(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 October 2001 (11.10.2001)

PCT

(10) International Publication Number
WO 01/75564 A2

(51) International Patent Classification[7]: G06F 1/00

(21) International Application Number: PCT/US01/08891

(22) International Filing Date: 21 March 2001 (21.03.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/541,108 31 March 2000 (31.03.2000) US

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **HERBERT, Howard, C.** [US/US]; 16817 South 1st Drive, Phoenix, AZ 85045 (US). **GRAWROCK, David, W.** [US/US]; 8285 S.W. 184th Avenue, Aloha, OR 97007 (US). **ELLISON, Carl, M.** [US/US]; 181 N.W. 28th Avenue, Portland, OR 97210 (US). **GOLLIVER, Roger, A.** [US/US]; 16185 S.W. Night Hawk Drive, Beaverton, OR 97007 (US). **LIN, Derrick, C.** [US/US]; 1737 Oakwood Drive, San Mateo, CA 94403 (US). **MCKEEN, Francis, X.** [US/US]; 10612 N.W. LeMans Court, Portland, OR 97229 (US). **NEIGER, Gilbert** [US/US]; 2424 N.E. 11th Avenue, Portland, OR 97212 (US). **RENERIS, Ken** [US/US]; 8 Red Gap Road, Wilbraham, MA 01095 (US). **SUTTON, James, A.** [US/US]; 20205 N.W. Paulina Drive, Portland, OR 97229 (US). **THAKKAR, Shreekant, S.** [GB/US]; 150 S.W. Moonridge Place, Portland, OR 97225 (US). **MITTAL, Millind** [US/US]; 800 E. Charleston Road #29, Palo Alto, CA 94303 (US).

(74) Agents: **MALLIE, Michael, J.** et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 01/75564 A2

(54) Title: PLATFORM AND METHOD FOR REMOTE ATTESTATION OF A PLATFORM

(57) Abstract: In one embodiment, a method of remote attestation for a special mode of operation. The method comprises storing an audit log within protected memory of a platform. The audit log is a listing of data representing each of a plurality of IsoX software modules loaded into the platform. The audit log is retrieved from the protected memory in response to receiving a remote attestation request from a remotely located platform. Then, the retrieved audit log is digitally signed to produce a digital signature for transfer to the remotely located platform.

PLATFORM AND METHOD FOR REMOTE ATTESTATION OF A PLATFORM

BACKGROUND

1. Field

This invention relates to the field of platform security.

2. Background

Advances in microprocessor and communication technologies with a platform have opened up many opportunities for applications that go beyond the traditional ways of doing business. Electronic commerce (e-commerce) and business-to-business (B2B) transactions are now becoming popular, reaching the global markets at a fast rate. Unfortunately, while modern microprocessor technology provides users with convenient and efficient methods of doing business, communicating and transacting, this technology fails to support remote attestation. Remote attestation is a technique for ascertaining the operating state of a remotely located platform in a generally secure manner. By ascertaining the operating state of the platform prior to conducting e-commerce or B2B transactions with that platform, the user is imparted with greater confidence in the security of the transaction.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

Figure 1A is a diagram illustrating an embodiment of the logical operating architecture for the IsoX™ architecture of the platform.

Figure 1B is an illustrative diagram showing the accessibility of various elements in the operating system and the processor according to one embodiment of the invention.

Figure 1C is a first block diagram of an illustrative embodiment of a platform utilizing the present invention.

Figure 2 is a flowchart of the illustrative operations of the platform to generate an embodiment of the protected audit log.

Figure 3 is a block diagram of an illustrative embodiment of a remote attestation unit employed in the processor of Figure 1C to obtain a protected copy of the audit log.

Figure 4 is a block diagram of an illustrative embodiment of a remote attestation unit employed in the chipset of Figure 1C to obtain a protected copy of the audit log external to the chipset.

Figure 5 is a block diagram of an illustrative embodiment of a remote attestation unit employed in the chipset of Figure 1C to obtain a protected copy of the audit log internal to the chipset.

Figure 6 is a block diagram of an illustrative embodiment of a remote attestation unit employed in the fixed token of Figure 1C to obtain a protected copy of the audit log.

Figure 7 is a block diagram of an illustrative embodiment of a remote attestation unit employed in the removable token of Figure 1C to obtain a protected copy of the audit log.

DESCRIPTION

The present invention relates to a platform and method for remote attestation of a platform. Remote attestation may be conducted when the platform is operating in a special mode of operation. An example of this special mode includes a processor isolated execution "IsoX" mode as described below. More specifically, a processor executing in IsoX mode utilizes hardware-protected keying material that is cryptographically unique to produce a digital signature that includes information concerning the operating environment of the platform. The hardware that provides protection of the keying material, referred to herein as a "remote attestation unit" (RAU),

may be integrated in a core logic device (e.g., a processor or a chipset component) or a non-core logic device (e.g., token).

In the following description, certain terminology is used to discuss features of the present invention. For example, a "platform" includes components that perform different functions on stored information. Examples of a platform include, but are not limited or restricted to a computer (e.g., desktop, a laptop, a hand-held, a server, a workstation, etc.), desktop office equipment (e.g., printer, scanner, a facsimile machine, etc.), a wireless telephone handset, a television set-top box, and the like. Examples of a "component" include hardware (e.g., an integrated circuit, etc.) and/or one or more software modules. A "software module" is code that, when executed, performs a certain function. This code may include an operating system, an application, an applet or even a nub being a series of code instructions, possibly a subset of code from an applet. A "link" is broadly defined as one or more information-carrying mediums (e.g., electrical wire, optical fiber, cable, bus, or air in combination with wireless signaling technology) to establish a communication pathway. This pathway is deemed "protected" when it is virtually impossible to modify information routed over the pathway without detection.

In addition, the term "information" is defined as one or more bits of data, address, and/or control and a "segment" is one or more bytes of information. A "message" is a grouping of information, possibly packetized information. "Keying material" includes any information needed for a specific cryptographic algorithm such as a Digital Signature Algorithm. A "one-way function" is a function, mathematical or otherwise, that converts information from a variable-length to a fixed-length (referred to as a "hash value" or "digest"). The term "one-way" indicates that there does not readily exist an inverse function to recover any discernible portion of the original information from the fixed-length hash value. Examples of a hash function include MD5 provided by RSA Data Security of Redwood City, California, or Secure Hash Algorithm (SHA-1) as specified in a 1995 publication Secure Hash Standard FIPS 180-1 entitled "Federal Information Processing Standards Publication" (April 17, 1995).

I. Architecture Overview

In one embodiment, a platform utilizing the present invention may be configured with an isolated execution (IsoX™) architecture. The IsoX™ architecture includes logical and physical definitions of hardware and software components that interact directly or indirectly with an operating system of the platform. Herein, the operating system and a processor of the platform may have several levels of hierarchy, referred to as rings, which correspond to various operational modes. A "ring" is a logical division of hardware and software components that are designed to perform dedicated tasks within the platform. The division is typically based on the degree or level of privilege, namely the ability to make changes to the platform. For example, a ring-0 is the innermost ring, being at the highest level of the hierarchy. Ring-0 encompasses the most critical, privileged components. Ring-3 is the outermost ring, being at the lowest level of the hierarchy. Ring-3 typically encompasses user level applications, which are normally given the lowest level of privilege. Ring-1 and ring-2 represent the intermediate rings with decreasing levels of privilege.

Figure 1A is a diagram illustrating an embodiment of a logical operating architecture 50 of the IsoX™ architecture. The logical operating architecture 50 is an abstraction of the components of the operating system and processor. The logical operating architecture 50 includes ring-0 10, ring-1 20, ring-2 30, ring-3 40, and a processor nub loader 52. Each ring in the logical operating architecture 50 can operate in either (i) a normal execution mode or (ii) an IsoX mode. The processor nub loader 52 is an instance of a processor executive (PE) handler.

Ring-0 10 includes two portions: a normal execution Ring-0 11 and an isolated execution Ring-0 15. The normal execution Ring-0 11 includes software modules that are critical for the operating system, usually referred to as the "kernel". These software modules include a primary operating system 12 (e.g., kernel), software drivers 13, and hardware drivers 14. The isolated execution Ring-0 15 includes an operating system (OS) nub 16 and a processor nub 18 as described below. The OS nub 16 and the processor nub 18 are instances of an OS executive (OSE) and processor executive (PE), respectively. The OSE and the PE are part of executive entities that operate in a

protected environment associated with the isolated area 70 and the IsoX mode. The processor nub loader 52 is a bootstrap loader code that is responsible for loading the processor nub 18 from the processor or chipset into an isolated area as will be explained later.

Similarly, ring-1 20, ring-2 30, and ring-3 40 include normal execution ring-1 21, ring-2 31, ring-3 41, and isolated execution ring-1 25, ring-2 35, and ring-3 45, respectively. In particular, normal execution ring-3 includes N applications 42_1-42_N and isolated execution ring-3 includes M applets 46_1-46_M (where "N" and "M" are positive whole numbers).

One concept of the IsoX™ architecture is the creation of an isolated region in the system memory, which is protected by components of the platform (e.g., the processor and chipset). This isolated region, referred to herein as an "isolated area," may also be in cache memory that is protected by a translation look aside (TLB) access check. Access to this isolated area is permitted only from a front side bus (FSB) of the processor, using special bus cycles (referred to as "isolated read and write cycles") issued by the processor executing in IsoX mode.

It is contemplated that links dedicated to solely support special cycles during remote attestation (referred to as "attestation cycles") may be employed within the platform. These attestation cycles may be based on the isolated read and write cycles or may be independent from the isolated read and write cycles. In lieu of dedicated links, shared links may be employed within the platform to support remote attestation. Examples of these shared links include a Peripheral Component Interconnect (PCI) bus, an accelerated graphics port (AGP) bus, an Industry Standard Architecture (ISA) bus, a Universal Serial Bus (USB) bus and the like. The attestation cycles are issued to prove locality, namely that a device with the keying material and a signing engine is accessing information (e.g., an audit log) stored in protected memory within the platform. This mitigates the threat of software simulating the retrieval of the audit log for example.

The IsoX mode is initialized using a privileged instruction in the processor, combined with the processor nub loader 52. The processor nub loader 52 verifies and

loads a ring-0 nub software module (e.g., processor nub 18) into the isolated area. For security purposes, the processor nub loader 52 is non-modifiable, tamper-resistant and non-substitutable. In one embodiment, the processor nub loader 52 is implemented in read only memory (ROM).

One task of the processor nub 18 is to verify and load the ring-0 OS nub 16 into the isolated area. The OS nub 16 provides links to services in the primary operating system 12 (e.g., the unprotected segments of the operating system), provides page management within the isolated area, and has the responsibility for loading ring-3 application modules 45, including applets 46_1 to 46_M, into protected pages allocated in the isolated area. The OS nub 16 may also support paging of data between the isolated area and ordinary (e.g., non-isolated) memory. If so, then the OS nub 16 is also responsible for the integrity and confidentiality of the isolated area pages before evicting the page to the ordinary memory, and for checking the page contents upon restoration of the page.

Referring now to Figure 1B, a diagram of the illustrative elements associated with the operating system 10 and the processor for one embodiment of the invention is shown. For illustration purposes, only elements of ring-0 10 and ring-3 40 are shown. The various elements in the logical operating architecture 50 access an accessible physical memory 60 according to their ring hierarchy and the execution mode.

The accessible physical memory 60 includes an isolated area 70 and a non-isolated area 80. The isolated area 70 includes applet pages 72 and nub pages 74. The non-isolated area 80 includes application pages 82 and operating system pages 84. The isolated area 70 is accessible only to components of the operating system and processor operating in the IsoX mode. The non-isolated area 80 is accessible to all elements of the ring-0 operating system and processor.

The normal execution ring-0 11 including the primary OS 12, the software drivers 13, and the hardware drivers 14, can access both the OS pages 84 and the application pages 82. The normal execution ring-3, including applications 42_1 to 42_N, can access

only to the application pages 82. Both the normal execution ring-0 11 and ring-3 41, however, cannot access the isolated area 70.

The isolated execution ring-0 15, including the OS nub 16 and the processor nub 18, can access to both of the isolated area 70, including the applet pages 72 and the nub pages 74, and the non-isolated area 80, including the application pages 82 and the OS pages 84. The isolated execution ring-3 45, including applets 46_1 to 46_M, can access only to the application pages 82 and the applet pages 72. The applets 46_1 to 46_M reside in the isolated area 70.

Referring to Figure 1C, a block diagram of an illustrative embodiment of a platform utilizing the present invention is shown. In this embodiment, platform 100 comprises a processor 110, a chipset 120, a system memory 140 and peripheral components (e.g., tokens 180/182 coupled to a token link 185 and/or a token reader 190) in communication with each other. It is further contemplated that the platform 100 may contain optional components such as a non-volatile memory (e.g., flash) 160 and additional peripheral components. Examples of these additional peripheral components include, but are not limited or restricted to a mass storage device 170 and one or more input/output (I/O) devices 175. For clarity, the specific links for these peripheral components (e.g., PCI bus, AGP bus, ISA bus, USB bus, wireless transmitter/receiver combinations, etc.) are not shown.

In general, the processor 110 represents a central processing unit of any type of architecture, such as complex instruction set computers (CISC), reduced instruction set computers (RISC), very long instruction word (VLIW), or hybrid architecture. In one embodiment, the processor 110 includes multiple logical processors. A "logical processor," sometimes referred to as a thread, is a functional unit within a physical processor having an architectural state and physical resources allocated according to a specific partitioning functionality. Thus, a multi-threaded processor includes multiple logical processors. The processor 110 is compatible with the Intel Architecture (IA) processor, such as a PENTIUM® series, the IA-32™ and IA-64™ . It will be appreciated by those skilled in the art that the basic description and operation of the processor 110 applies to either a single processor platform or a multi-processor platform.

The processor 110 may operate in a normal execution mode or an IsoX mode. In particular, an isolated execution circuit 115 provides a mechanism to allow the processor 110 to operate in an IsoX mode. The isolated execution circuit 115 provides hardware and software support for the IsoX mode. This support includes configuration for isolated execution, definition of the isolated area, definition (e.g., decoding and execution) of isolated instructions, generation of isolated access bus cycles, and generation of isolated mode interrupts. In one embodiment, as shown in Figure 3, the RAU may be implemented as part of the processor 110.

As shown in Figure 1C, a host link 116 is a front side bus that provides interface signals to allow the processor 110 to communicate with other processors or the chipset 120. In addition to normal mode, the host link 116 supports an isolated access link mode with corresponding interface signals for isolated read and write cycles when the processor 110 is configured in the IsoX mode. The isolated access link mode is asserted on memory accesses initiated while the processor 110 is in the IsoX mode if the physical address falls within the isolated area address range. The isolated access link mode is also asserted on instruction pre-fetch and cache write-back cycles if the address is within the isolated area address range. The processor 110 responds to snoop cycles to a cached address within the isolated area address range if the isolated access bus cycle is asserted.

Herein, the chipset 120 includes a memory control hub (MCH) 130 and an input/output control hub (ICH) 150 described below. The MCH 130 and the ICH 150 may be integrated into the same chip or placed in separate chips operating together. In another embodiment, as shown in Figure 4, the RAU may be implemented as part of the chipset 120.

With respect to the chipset 120, a MCH 130 provides control and configuration of memory and input/output devices such as the system memory 140 and the ICH 150. The MCH 130 provides interface circuits to recognize and service attestation cycles and/or isolated memory read and write cycles. In addition, the MCH 130 has memory range registers (e.g., base and length registers) to represent the isolated area in the system memory 140. Once configured, the MCH 130 aborts any access to the isolated area when the isolated access link mode is not asserted.

The system memory 140 stores code and data. The system memory 140 is typically implemented with dynamic random access memory (DRAM) or static random access memory (SRAM). The system memory 140 includes the accessible physical memory 60 (shown in Figure 1B). The accessible physical memory 60 includes the isolated area 70 and the non-isolated area 80 as shown in Figure 1B. The isolated area 70 is the memory area that is defined by the processor 110 when operating in the IsoX mode. Access to the isolated area 70 is restricted and is enforced by the processor 110 and/or the chipset 120 that integrates the isolated area functionality. The non-isolated area 80 includes a loaded operating system (OS). The loaded OS 142 is the portion of the operating system that is typically loaded from the mass storage device 170 via some boot code in a boot storage such as a boot read only memory (ROM). Of course, the system memory 140 may also include other programs or data which are not shown.

As shown in Figure 1C, the ICH 150 supports isolated execution in addition to traditional I/O functions. In this embodiment, the ICH 150 comprises at least the processor nub loader 52 (shown in Figure 1A), a hardware-protected memory 152, an isolated execution logical processing manager 154, and a token link interface 158. For clarity, only one ICH 150 is shown although platform 100 may be implemented with multiple ICHs. When there are multiple ICHs, a designated ICH is selected to control the isolated area configuration and status. This selection may be performed by an external strapping pin. As is known by one skilled in the art, other methods of selecting can be used.

The processor nub loader 52, as shown in Figures 1A and 1C, includes a processor nub loader code and its hash value (or digest). After being invoked by execution of an appropriated isolated instruction (e.g., ISO_INIT) by the processor 110, the processor nub loader 52 is transferred to the isolated area 70. Thereafter, the processor nub loader 52 copies the processor nub 18 from the non-volatile memory 160 into the isolated area 70, verifies and places a representation of the processor nub 18 (e.g., a hash value) into the protected memory 152. Herein, the protected memory 152 is implemented as a memory array with single write, multiple read capability. This non-modifiable capability is controlled by logic or is part of the inherent nature of the

memory itself. For example, as shown, the protected memory 152 may include a plurality of single write, multiple read registers.

As shown in Figures 1C and 2, the protected memory 152 is configured to support an audit log 156. An "audit log" 156 is information concerning the operating environment of the platform 100; namely, a listing of data that represents what information has been successfully loaded into the system memory 140 after power-on of the platform 100. For example, the representative data may be hash values of each software module loaded into the system memory 140. These software modules may include the processor nub 18, the OS nub 16, and/or any other critical software modules (e.g., ring-0 modules) loaded into the isolated area 70. Thus, the audit log 156 can act as a fingerprint that identifies information loaded into the platform (e.g., the ring-0 code controlling the isolated execution configuration and operation), and is used to attest or prove the state of the current isolated execution.

In another embodiment, both the protected memory 152 and unprotected memory (e.g., a memory array in the non-isolated area 80 of the system memory 140 of Figure 1C) may collectively provide a protected audit log 156. The audit log 156 is stored in the memory array while information concerning the state of the audit log 156 (e.g., a total hash value for the representative data within the audit log 156) is stored in the protected memory 152.

Referring still to Figure 1C, the non-volatile memory 160 stores non-volatile information. Typically, the non-volatile memory 160 is implemented in flash memory. The non-volatile memory 160 includes the processor nub 18 as described above. Additionally, the processor nub 18 may also provide application programming interface (API) abstractions to low-level security services provided by other hardware and may be distributed by the original equipment manufacturer (OEM) or operating system vendor (OSV) via a boot disk.

The mass storage device 170 stores archive information such as code (e.g., processor nub 18), programs, files, data, applications (e.g., applications 42_1-42_N), applets (e.g., applets 46_1 to 46_M) and operating systems. The mass storage device 170 may

include a compact disk (CD) ROM 172, a hard drive 176, or any other magnetic or optic storage devices. The mass storage device 170 also provides a mechanism to read platform-readable media. When implemented in software, the elements of the present invention are stored in a processor readable medium. The "processor readable medium" may include any medium that can store or transfer information. Examples of the processor readable medium include an electronic circuit, a semiconductor memory device, a read only memory (ROM), a flash memory, an erasable programmable ROM (EPROM), a fiber optic medium, a radio frequency (RF) link, and any platform readable media such as a floppy diskette, a CD-ROM, an optical disk, a hard disk, etc.

In communication with the platform 100, I/O devices 175 include stationary or portable user input devices, each of which performs one or more I/O functions. Examples of a stationary user input device include a keyboard, a keypad, a mouse, a trackball, a touch pad, and a stylus. Examples of a portable user input device include a handset, beeper, hand-held (e.g., personal digital assistant) or any wireless device. The I/O devices 175 enable remote attestation of the platform 100 as described below.

The token link 185 provides an interface between the ICH 150 and a fixed token 180 (e.g., a motherboard token) and/or a token reader 190 in communication with a removable token 182 having characteristics similar to a smart card. In general, both types of tokens are devices that perform dedicated I/O functions. For embodiments shown in Figures 6 and 7, tokens 180 and/or 182 include keying material (e.g., unique cryptographic identifier such as a public/private key pair) and functionality to digitally sign the audit log (or a representation thereof) with the private key of the key pair. The token link interface 158 in the ICH 150 provides a logical coupling between the token link 185 and the ICH 150 and supports remote attestation for recovery of the contents of the audit log 156.

II. Generating and Utilizing a Protected Audit Log

Referring now to Figure 2, a flowchart of the illustrative operations of the platform to generate an embodiment of the protected audit log is shown. After power-on of the platform, segments of information are loaded into the system memory for

processing by a processor (block 200). Examples of these segments of information include the processor nub and the OS nub. Concurrent with the loading of the segments of information into the system memory, copies of each segment of the information undergo a cryptographic hash operation to produce a hash value of the segments. These hash values form an audit log stored in protected memory (blocks 205 and 210). In one embodiment, as shown in Figure 1C, the protected memory is implemented within the ICH. The memory is deemed "protected" when the contents of the memory are readable and non-modifiable as described above. As subsequent segments of information are selected for storage into the audit log, their hash values are appended to the audit log behind the previously computed hash values (block 215). It is contemplated that only hash values of selected nubs may be stored in the audit log.

III. Remote Attestation

A. Commencement of Remote Attestation

In one embodiment, remote attestation is initiated by issuing an attestation request. The attestation request can originate from a remote source or from an agent, local to the platform, which may or may not be acting as a proxy for the remote source. Normally, the attestation request comprises a primary query and/or one or more optional secondary queries. Each query causes the issuance of the attestation cycles, which are designed to retrieve contents of the audit log. At a minimum, the contents of the audit log may be used to verify the integrity of IsoX™ processor and the OS nub of the platform. The secondary query retrieves, in addition to the audit log, a hash value of a selected IsoX applet loaded by the platform in order to verify the integrity of the applet. The hash value of the applet is generated on the fly by the OS nub. This avoids the need to store each and every loaded applet in the audit log. For primary queries, the RAU creates a message that may include the audit log, a digital signature covering the audit log, and one or more digital certificates for the RAU keying material and returns the message to the requestor. For secondary queries, the RAU creates a message that may include the applet hash, the audit log, a digital signature covering the applet hash and audit log, and one or more digital certificates for the RAU keying material and returns the message to the requestor to retrieve different information cited above.

B. Processor Integrated RAU

Referring now to Figure 3, the RAU 300 is integrated into the processor 110. The processor 110 is executing local code. Upon detection of an attestation request, the processor 110 establishes a communication pathway with a component 310 responsible for storing the audit log 156. More specifically, in one embodiment, the local code executes a physical instruction in response to an attestation request. The physical instruction, when executed by the processor 110, causes the issuance of attestation cycles by the processor 110 for reading contents of the audit log 156.

For illustrative sake, the component 310 may be the ICH 150 of Figure 1C, although other components within the platform 100 may be used. The communications between the processor 110 and component 310 are through one or more links such as a first link 310 and a second link 320. These links 310 and 320 may be configured as dedicated links for handling attestation cycles or shared links (e.g., host link, PCI bus, etc.) enhanced to handle the attestation cycles. These attestation cycles signal the component 310 to accept reads of the audit log 156.

Upon receiving the audit log 156, the RAU 300 in the processor 110 produces a digital signature 330 by digitally signing the audit log 156 with the keying material 340 (e.g., a pre-stored private key). The audit log 156, digital signature 330, and possibly digital certificates from the RAU keying material and packetized and sent as a message by the RAU 300 to the requestor or to an area 350 accessible to the local code.

Of course, it is contemplated that if the audit log 156 is stored in unprotected memory, the ICH 150 may include a component (not shown) to verify that the contents of the audit log 156 have not been modified before releasing the audit log 156 to the processor 110. This may be accomplished by the component 310 generating a hash value of the audit log 156 recovered from unprotected memory and comparing the hash value to the total hash value stored in protected memory.

As an optional embodiment, the user may want to control when the keying material 340 is used. For example, the platform may issue a request message via a communications device 360 to a user opt-in device 380 over a protected communication

path. In one embodiment, the communications device 360 is coupled to the token bus 185 and is employed with a wireless receiver 365 and a wireless transmitter 370 (collectively referred to herein as a "wireless transceiver"). The wireless receiver and transmitter 365 and 370 are used to establish and maintain direct communications with the user opt-in device 380. Of course, the user opt-in device 380 may be coupled to communications device 360 via any link type.

Upon receipt of the request message, the communications device 360 issues a message to the user opt-in device 380 which enables the user to affirm his or her desire to release the keying material 340 for generation of the digital signature 330. Based on an input by the user or lack thereof (e.g., depression of a key associated with user opt-in device 380, inaction by the user, etc.), a response message is returned to the communications device 360, which routes the contents of the response message to the RAU 300 over a protected communication path. Upon receipt of the response message, the RAU 300 proceeds with the generation of the digital signature 330 and/or digital certificates for the RAU keying material and placement in the area 350 accessible to the local code if use of the keying material 340 is authorized by the user.

C. Chipset Integrated RAU

Referring now to Figure 4, the RAU 300 is integrated into a core logic device 400. As shown, the processor 110 is executing local code. Upon detection of an attestation request, the core logic device 400 establishes a communication pathway with a component 420 responsible for storing the audit log 156. More specifically, in one embodiment, the local code sends a message to core logic device 400 based on an attestation request. The message causes the core logic device 400 to issue attestation cycles for reading contents of the audit log 156.

For example, in response to the attestation request, the core logic device 400 routes the attestation cycles to the component 420 via link 430 to allow contents of the stored audit log 156 to be read. Link 430 may be dedicated to support remote attestation or support multiple functions inclusive of attestation cycles generated by the core logic device 400. Upon receiving the contents of the stored audit log 156, the core logic

device 400 that contains the RAU 300 generates a digital signature 330 for the audit log 156 (as described above) and writes the digital signature 330 into an area accessible to the local code.

However, as shown in Figure 5, if the core logic device 400 also contains the audit log 156, internal signals 450 within the core logic device 400 are used to allow the RAU 300 to access the audit log 156. Again, upon receiving the contents of the audit log 156, the RAU 300 of the core logic device 400 generates the digital signature 330 of the audit log and possibly one or more digital certificates for the RAU keying material (not shown). This information is provided as a message to the requestor or written into the area accessible to the local code.

As an optional embodiment, the user may want to control when the keying material 340 is used. For example, the platform may issue a request message 470 via a communications device 460 to a user opt-in device 490 over a protected communication path. In one embodiment, the communications device 460 is coupled to the token bus 185 and is employed with a wireless transceiver 465 in order to establish and maintain direct communications with the user opt-in device 490.

In response to receiving the request message 470, the communications device 460 issues a message to the user opt-in device 490, which solicits the user to affirm his or her desire to release the keying material 340 for generation of the digital signature 330. Based on an input by the user or lack thereof (e.g., depression of a key associated with the user opt-in device 490, inaction by the user, etc.), a response message 480 is returned to the communications device 460, which routes the contents of the response message 480 to the RAU 300 of the core logic device 400 over a protected communication path. Upon receipt of the response message 480, the RAU 300 proceeds with the generation of the digital signature 330 and possibly digital certificates as described above and placement in the area accessible to the local code if use of the keying material 340 is authorized by the user.

D. Fixed Token Integrated RAU

Referring now to Figure 6, if the RAU 300 is integrated in the fixed token 180, the fixed token 180 communicates with a component (e.g., ICH 150) holding the audit log 156 over the token link 185. The functionality of token link 185 may be enhanced to support attestation cycles that are only generated by the fixed token 180 when remote attestation is being requested. These attestation cycles are routed to the ICH 150 to request acceptance of reads to the audit log 156. Upon receiving the contents of the audit log 156, the RAU 300 implemented in the fixed token 180 generates a digital signature 330 by digitally signing the audit log 156 with keying material 340 stored in the RAU 300. Thereafter, the RAU 300 writes the digital signature 330 and possibly digital certificates for keying material 340 to the requestor or into an area accessible to the local code.

As an optional embodiment, the user may want to control when the keying material 610 stored in the RAU 300 is used. For example, the user may be prompted to affirm his or her desire to release the keying material 340 for generation of the digital signature 330. The prompt may be accomplished, for example, through transmission of a message 620 via a wireless transceiver 630 situated in the token 180. Affirmation of a desire to release the keying material 340 may be made by either (1) transmitting a return message 640 from a user opt-in device to the token 180 as shown or (2) entering access information via a user opt-in device (not shown) physically connected to the token 180, for example. Thereafter, the RAU 300 proceeds with the generation of the digital signature 330 and/or digital certificate(s) for the keying material 340. Then, this information along with the audit log 156 are sent to the requestor or placed in the area accessible to the local code if use of the keying material 340 has been authorized by the user. Of course, opt-in messages 620 and 640 may be routed through the I/O device 175 provided the messages are protected.

E. Removable Token Integrated RAU

Referring now to Figure 7, if the RAU 300 is integrated in the removable token 182, the removable token 182 communicates with a component (e.g., ICH 150) holding

the audit log 156 over the token link 185. The functionality of token link 185 may be enhanced to support attestation cycles that are only generated by the token reader upon insertion or connection (i.e., wireless token) of removable token 182 when remote attestation is being requested. These attestation cycles are generated by the token reader 190 to the hardware storing the audit log 156 (e.g., ICH 150) to request acceptance of reads to the audit log 156. Upon receiving the contents of the audit log 156, the RAU 300 implemented in the removable token 182 generates the digital signature 330 by digitally signing the audit log 156 with keying material 340 stored in the RAU 300. Thereafter, the RAU 300 writes the digital signature 330 and/or digital certificate(s) for the keying material 340 into an area accessible to the local code.

As an optional embodiment, the user may want to control when the keying material 340 stored in the RAU 300 is used. For example, the user may be prompted to affirm his or her desire to release the keying material 340 for generation of the digital signature 330. The prompt may be accomplished, for example, through transmission of a message 720 via a wireless transceiver 730 situated in the token 182. Affirmation of a desire to release the keying material 340 may be made by either (1) transmitting a return message 740 from a user opt-in device (not shown) to the token 182 as shown or (2) entering access information via a user opt-in device physically connected to the token 182 (not shown) for example. Thereafter, the RAU 300 proceeds with the generation of the digital signature 330 and/or digital certificates for the keying material 340, routing through the token reader 190 and placement in the area accessible to the local code if use of the keying material 340 has been authorized by the user. Of course, opt-in messages 620 and 640 may be routed through the I/O device 175 provided the messages are protected.

While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.

CLAIMS

What is claimed is:

1. A platform comprising:

a processor including a remote attestation unit, the processor executing in one of a normal execution mode and an isolated execution mode;

a chipset to store an audit log; and

a link coupled to the processor and the chipset, the link to support predetermined bus cycles for the remote attestation unit to read contents of the audit log when a remote attestation request has been detected.

2. The platform of claim 1, wherein the remote attestation unit of the processor includes keying material.

3 The platform of claim 2, wherein the remote attestation unit of the processor includes a digital signature unit to digitally sign the audit log with the keying material.

4. The platform of claim 3, wherein the keying material within the remote attestation unit includes a private key.

5. The platform of claim 3, wherein the chipset includes:

a system memory including an isolated area and a non-isolated area;

a memory control hub coupled to system memory and the processor via a first link partially forming the link; and

an input/output control hub coupled to the memory control hub via a second link partially forming the link, the input/output control hub including single-write, multiple-read memory to store the audit log.

6. The platform of claim 5 further comprising a communications device coupled to the input/output control hub, the communications device enables communications with a user opt-in device.

7. The platform of claim 6, wherein the communications device includes a wireless transmitter and a wireless receiver to communicate with the user opt-in device.

8. The platform of claim 6, wherein the user opt-in device enables a user to control a stage of operation of the remote attestation by preventing the creation of the digital signature.

9. The platform of claim 2, wherein the remote attestation request includes a primary query.

10. The platform of claim 9, wherein the remote attestation unit returns a message to a requestor in response to the primary query, the message includes the audit log and at least a digital signature being the audit log digitally signed with the keying material.

11. The platform of claim 10, wherein the message further includes a digital certificate for the keying material.

12. The platform of claim 9, wherein the remote attestation request includes a secondary query.

13. The platform of claim 12, wherein the remote attestation unit returns a message to a requestor in response to the secondary query, the message includes a hash value of a selected applet, the audit log and a digital signature including the hash value and the audit log.

14. The platform of claim 13, wherein the message further includes a digital certificate for the keying material.

15. A platform comprising:

a component to contain an audit log; and

a device including a remote attestation unit to retrieve the audit log and digitally sign the audit log with keying material stored in the remote attestation unit, the audit log including representative data of software modules loaded within the platform after power-on.

16. The platform of claim 15 further comprising a processor to detect a remote attestation request and to issue cycles to the component to allow the device to access the audit log.

17. The platform of claim 15, wherein the device is a chipset.

18. The platform of claim 16 further comprising:

a chipset coupled to processor, the chipset including the component and a token link interface; and

a token link coupled to the chipset.

19. The platform of claim 15, wherein the device is a fixed token coupled to the token link.

20. The platform of claim 19, further comprising a user opt-in device in communication with the fixed token, the user opt-in device enables a user to cease operations of the remote attestation unit.

21. The platform of claim 18 further comprising a token reader coupled to the token link.

22. The platform of claim 21, wherein the device is a removable token in communication with the token reader.

23. The platform of claim 22, further comprising a user opt-in device in communication with the removable token, the user opt-in device enables a user to cease operations of the remote attestation unit.

24. A method comprising:

storing an audit log within protected memory of a platform, the audit log being a listing of data representing each of a plurality of IsoX software modules loaded into the platform;

retrieving the audit log from the protected memory in response to receiving a remote attestation request from a remotely located platform; and

digitally signing the audit log to produce a digital signature before transfer to the remotely located platform.

25. The method of claim 24, wherein the data representative of each of the plurality of software modules is a cryptographic hash value.

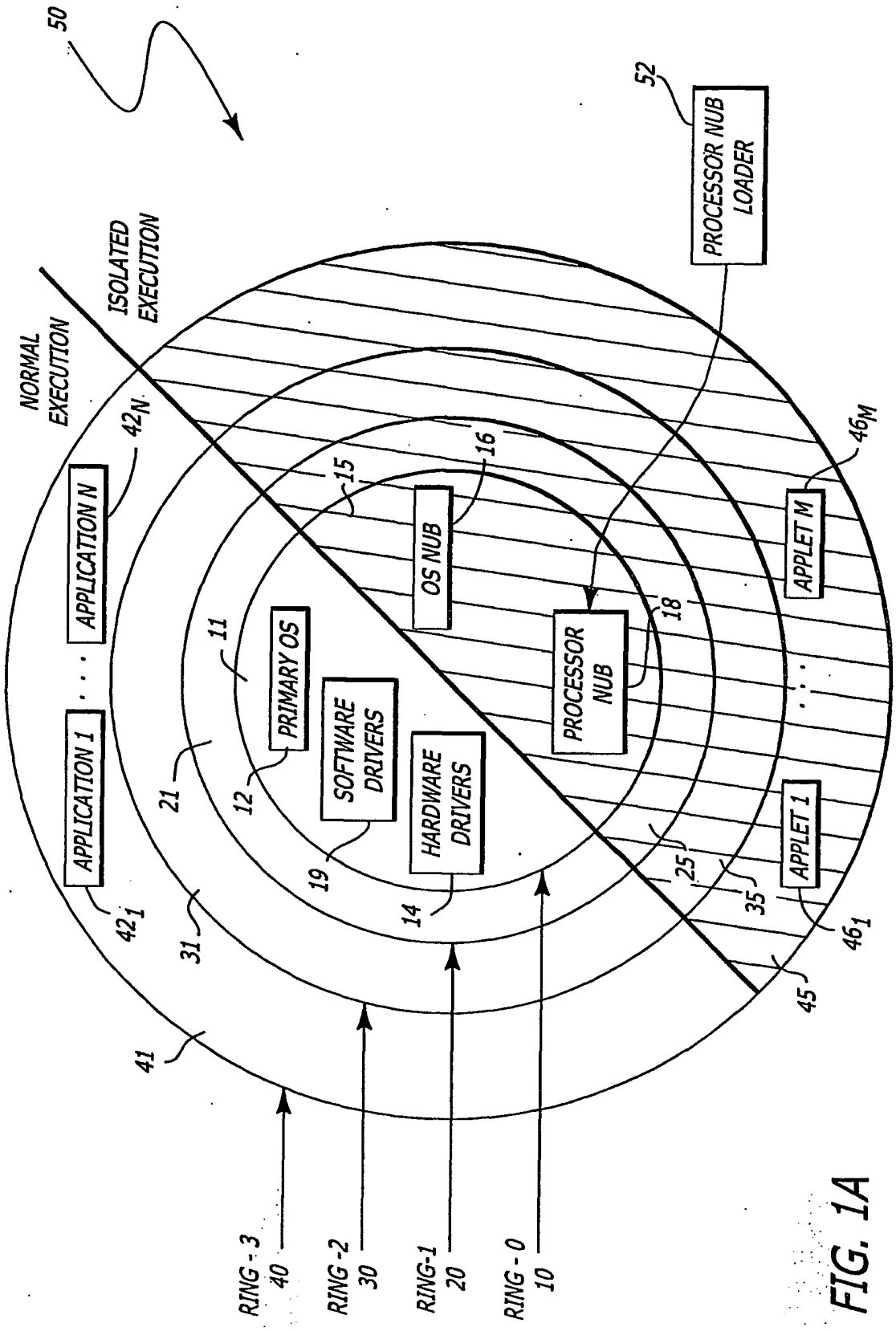

50
52
PROCESSOR NUB LOADER
NORMAL EXECUTION
ISOLATED EXECUTION
APPLICATION N
APPLICATION 1
OS NUB
PRIMARY OS
SOFTWARE DRIVERS
HARDWARE DRIVERS
PROCESSOR NUB
APPLET M
APPLET 1
RING - 3 40
RING -2 30
RING-1 20
RING - 0 10

FIG. 1A

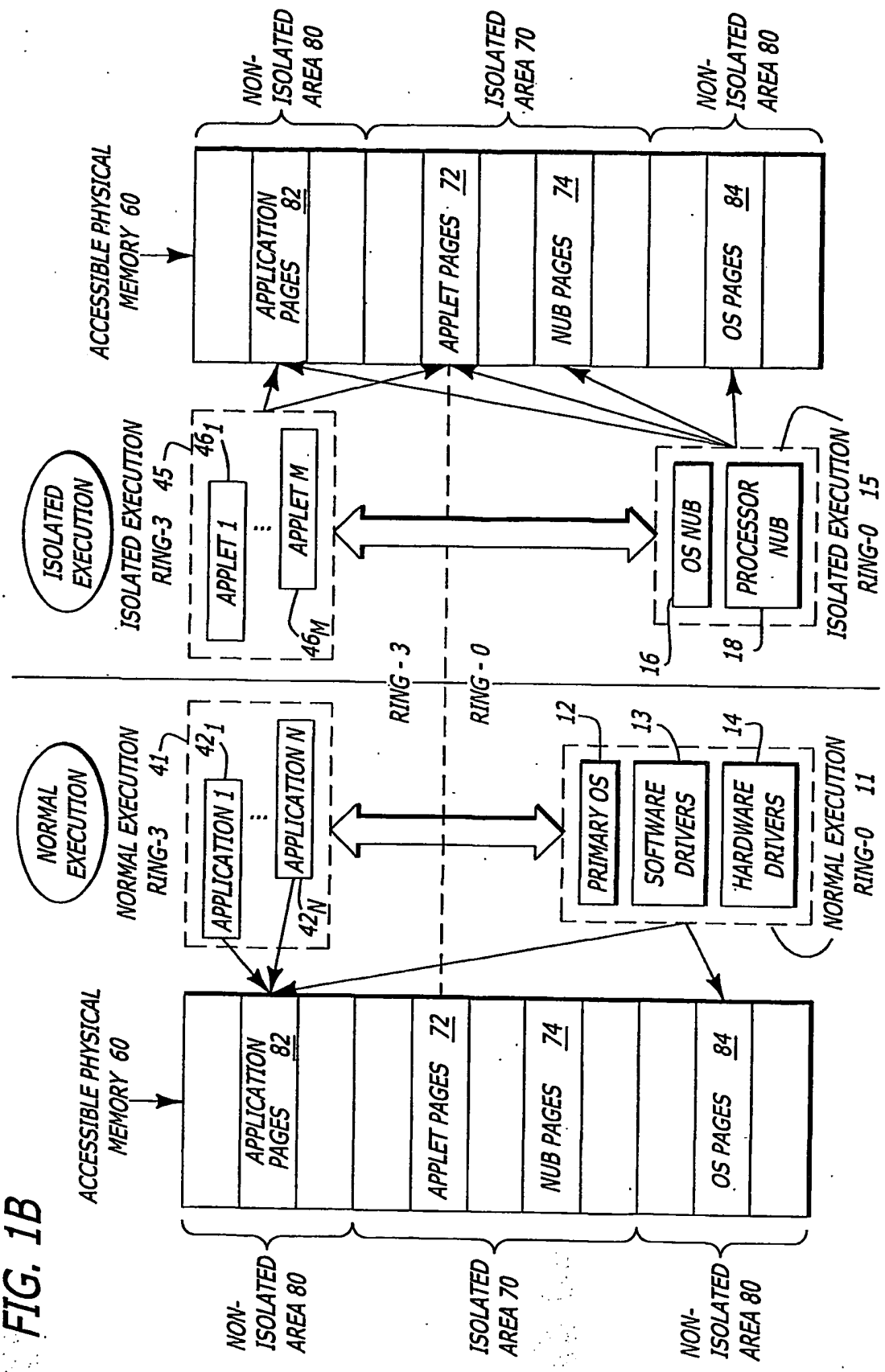
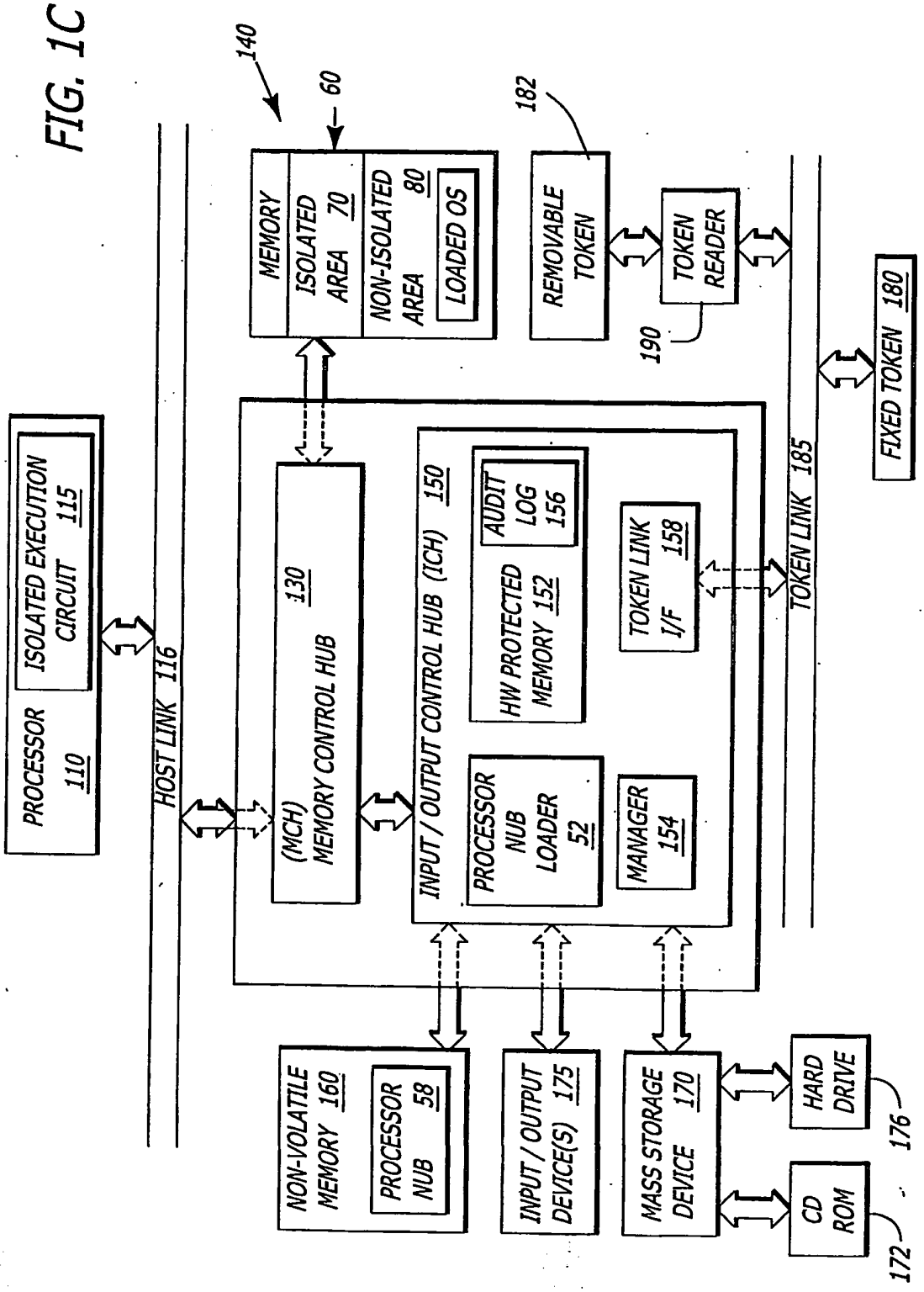

NON-ISOLATED AREA 80
ISOLATED AREA 70
NON-ISOLATED AREA 80
ACCESSIBLE PHYSICAL MEMORY 60
APPLICATION PAGES 82
APPLET PAGES 72
NUB PAGES 74
OS PAGES 84
ISOLATED EXECUTION
ISOLATED EXECUTION RING-3 45
APPLET 1 461
APPLET M 46M
OS NUB
PROCESSOR NUB
ISOLATED EXECUTION RING-0 15
16
18
RING - 3
RING - 0
NORMAL EXECUTION
NORMAL EXECUTION RING-3 41
APPLICATION 1 421
APPLICATION N 42N
12
13
14
PRIMARY OS
SOFTWARE DRIVERS
HARDWARE DRIVERS
NORMAL EXECUTION RING-0 11
ACCESSIBLE PHYSICAL MEMORY 60
APPLICATION PAGES 82
APPLET PAGES 72
NUB PAGES 74
OS PAGES 84
NON-ISOLATED AREA 80
ISOLATED AREA 70
NON-ISOLATED AREA 80

FIG. 1B

FIG. 1C


140
60
182
MEMORY
ISOLATED AREA 70
NON-ISOLATED AREA 80
LOADED OS
REMOVABLE TOKEN
TOKEN READER
190
FIXED TOKEN 180
PROCESSOR 110
ISOLATED EXECUTION CIRCUIT 115
HOST LINK 116
(MCH) MEMORY CONTROL HUB 130
INPUT / OUTPUT CONTROL HUB (ICH) 150
AUDIT LOG 156
HW PROTECTED MEMORY 152
TOKEN LINK I/F 158
TOKEN LINK 185
PROCESSOR NUB LOADER 52
MANAGER 154
NON-VOLATILE MEMORY 160
PROCESSOR NUB 58
INPUT / OUTPUT DEVICE(S) 175
MASS STORAGE DEVICE 170
HARD DRIVE
176
CD ROM
172

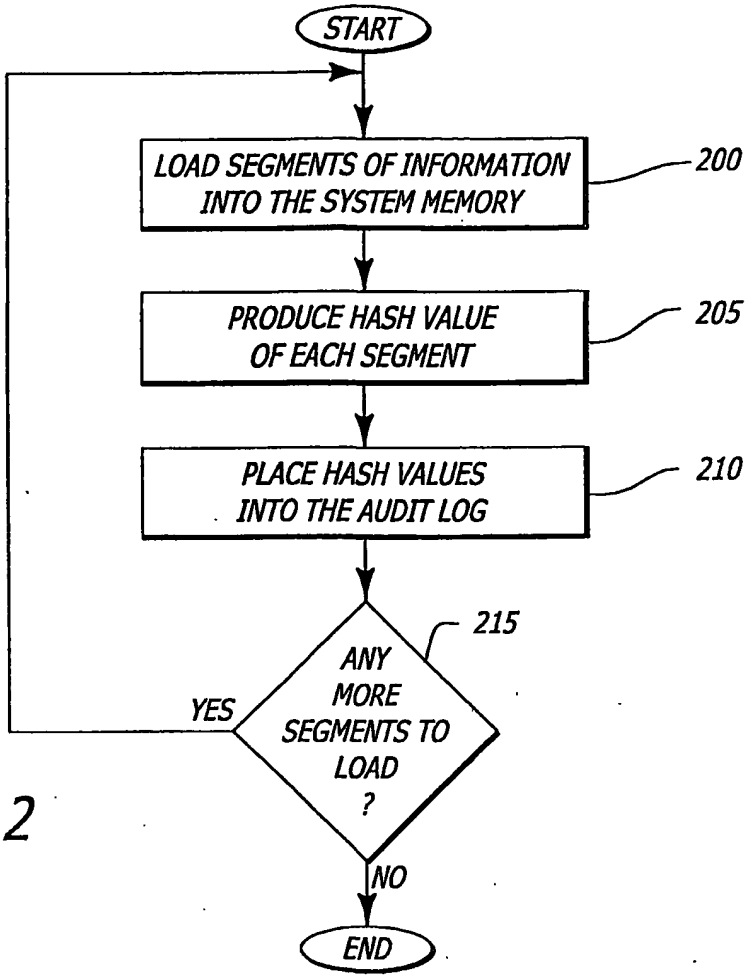

START
LOAD SEGMENTS OF INFORMATION INTO THE SYSTEM MEMORY
200
PRODUCE HASH VALUE OF EACH SEGMENT
205
PLACE HASH VALUES INTO THE AUDIT LOG
210
ANY MORE SEGMENTS TO LOAD ?
215
YES
NO
END

FIG. 2

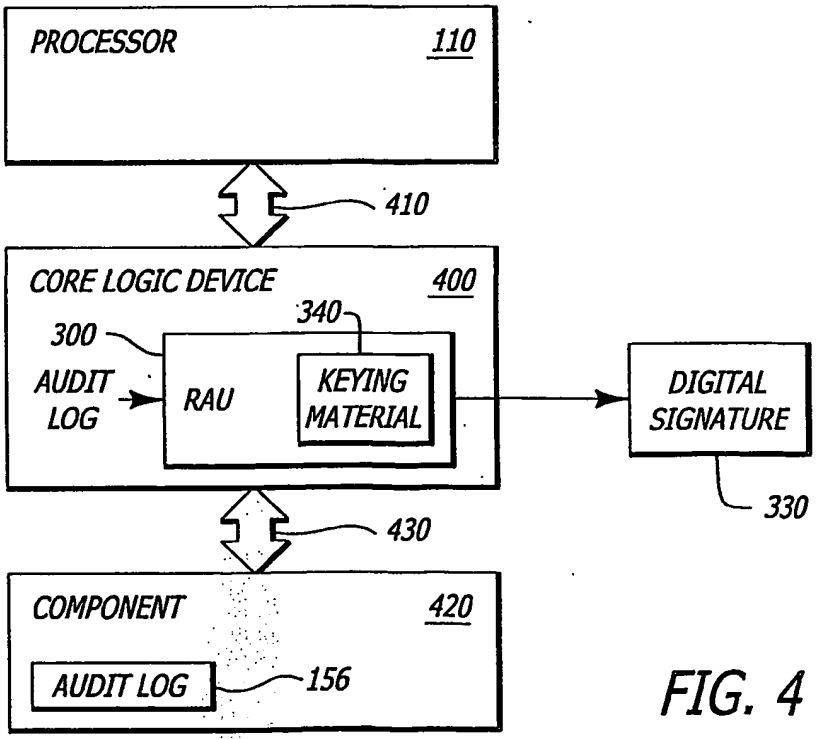
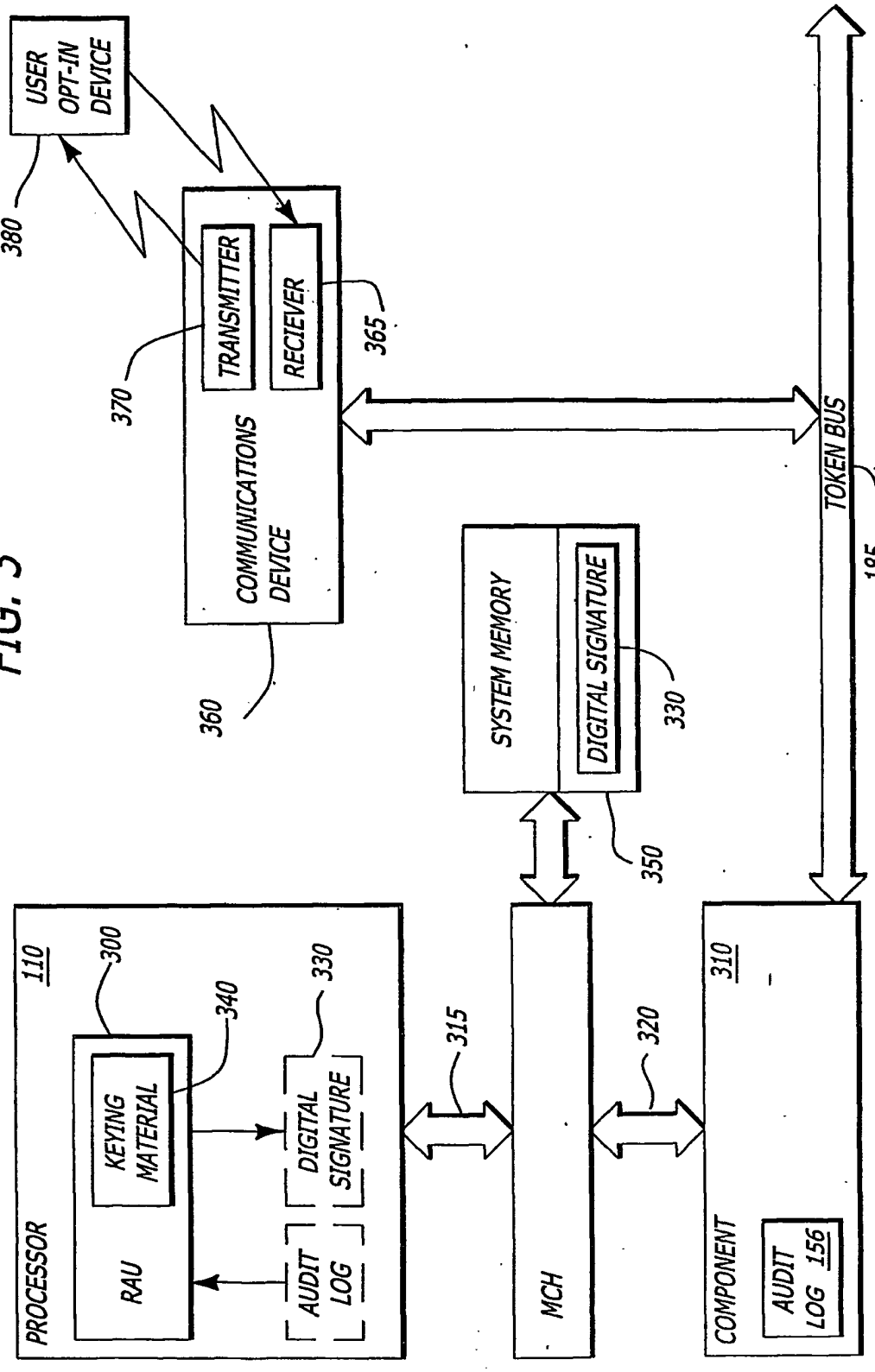

PROCESSOR
110
410
CORE LOGIC DEVICE
400
340
300
AUDIT LOG
RAU
KEYING MATERIAL
DIGITAL SIGNATURE
330
430
COMPONENT
420
AUDIT LOG
156

FIG. 4

FIG. 3


USER OPT-IN DEVICE
380
TRANSMITTER
RECIEVER
365
370
COMMUNICATIONS DEVICE
360
TOKEN BUS
185
SYSTEM MEMORY
DIGITAL SIGNATURE
330
350
PROCESSOR
110
300
340
330
KEYING MATERIAL
DIGITAL SIGNATURE
RAU
AUDIT LOG
315
MCH
320
310
COMPONENT
AUDIT LOG 156

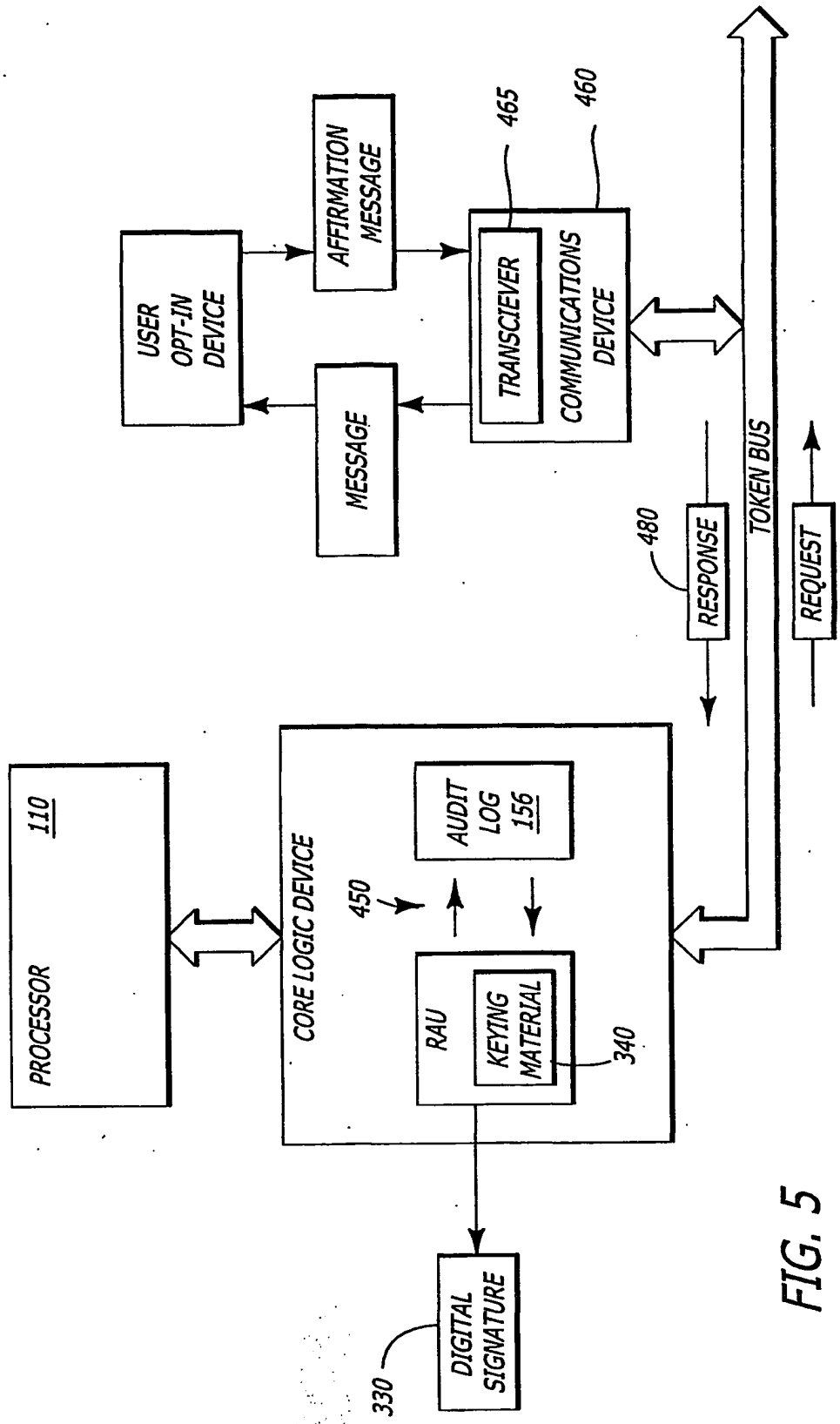

PROCESSOR
110
CORE LOGIC DEVICE
450
AUDIT LOG 156
RAU
KEYING MATERIAL
340
DIGITAL SIGNATURE
330
TOKEN BUS
REQUEST
RESPONSE
480
COMMUNICATIONS DEVICE
460
TRANSCIEVER
465
AFFIRMATION MESSAGE
MESSAGE
USER OPT-IN DEVICE

FIG. 5

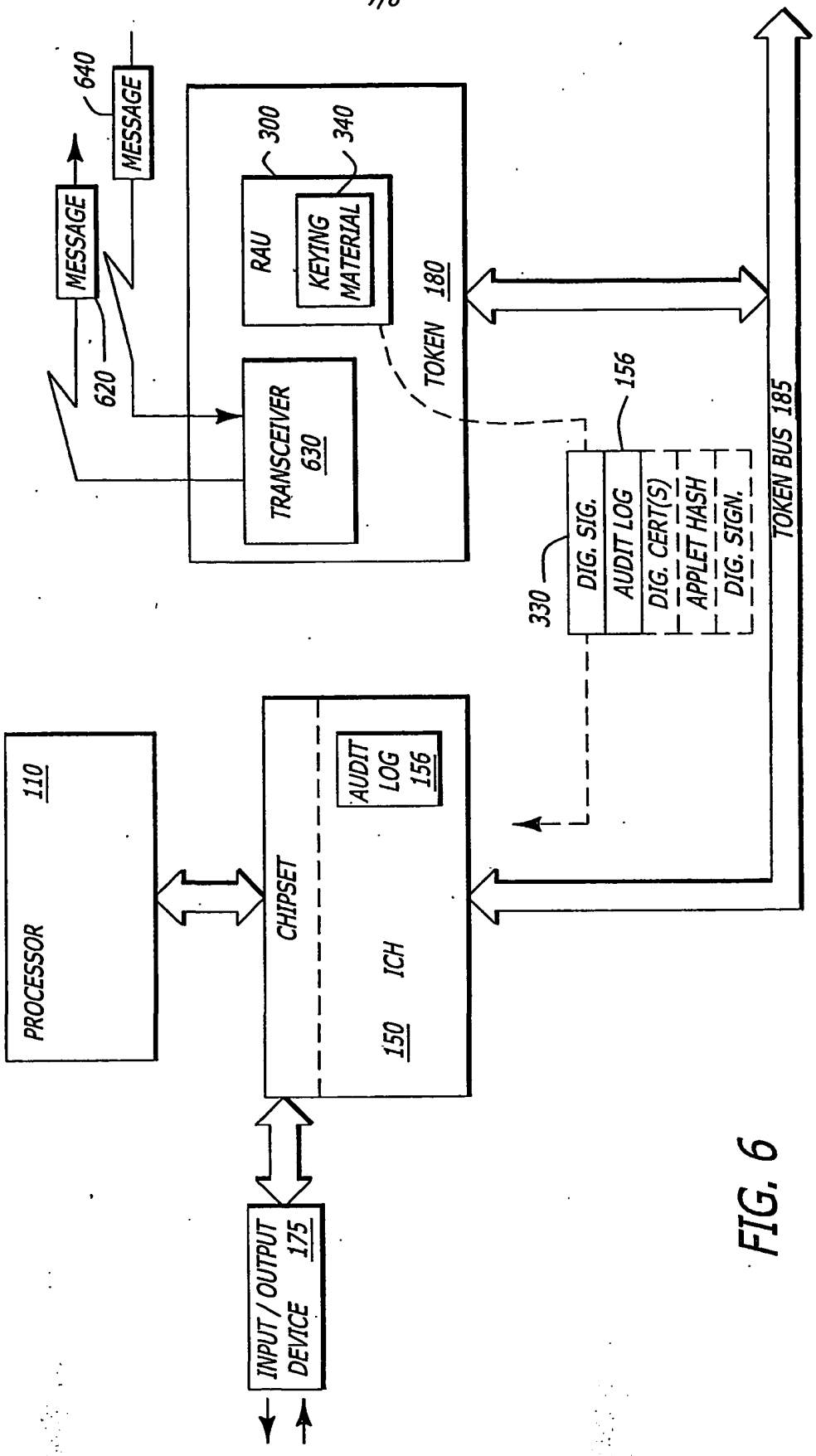

640
MESSAGE
MESSAGE
620
300
340
RAU
KEYING MATERIAL
TOKEN 180
TRANSCEIVER 630
156
330
DIG. SIG.
AUDIT LOG
DIG. CERT(S)
APPLET HASH
DIG. SIGN.
TOKEN BUS 185
PROCESSOR 110
CHIPSET
AUDIT LOG 156
ICH 150
INPUT / OUTPUT DEVICE 175

FIG. 6

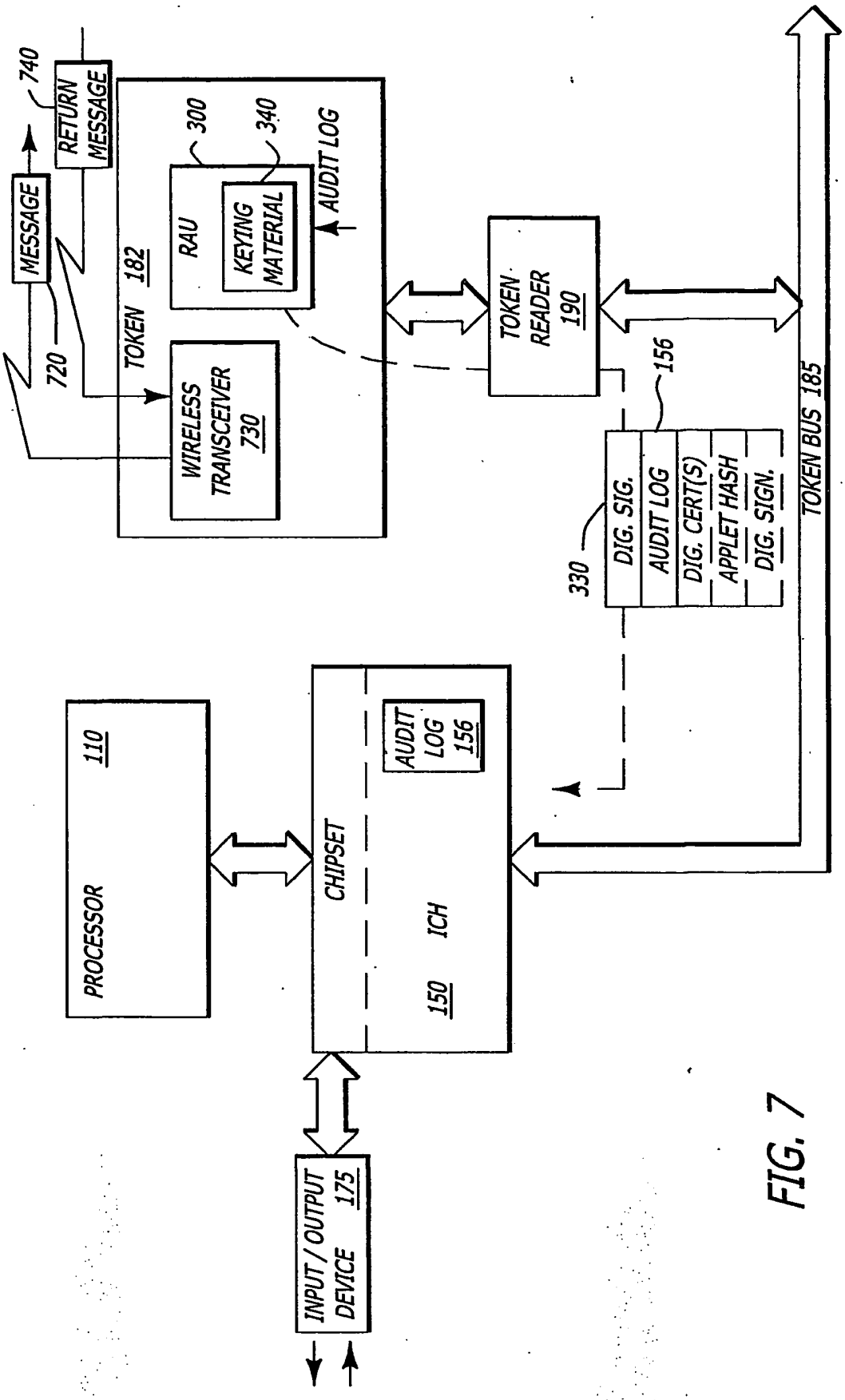

740
RETURN MESSAGE
MESSAGE
720
TOKEN 182
300
RAU
340
KEYING MATERIAL
AUDIT LOG
WIRELESS TRANSCEIVER 730
TOKEN READER 190
156
330
DIG. SIG.
AUDIT LOG
DIG. CERT(S)
APPLET HASH
DIG. SIGN.
TOKEN BUS 185
PROCESSOR 110
CHIPSET
AUDIT LOG 156
ICH 150
INPUT / OUTPUT DEVICE 175

FIG. 7

platform enforces a minimum set of requirements (protection of engines and/or services), and contains an entity that does security processing and hides that security processing from applications on the platform. A label that does this but is not associated with a given engine/service is called the "root label" of the platform. It can be used for signing when the platform meets the minimum requirements. Production of signatures incorporating other labels preferably depends upon the presence of the computing engine and/or service associated with the label. When merely advertising, a platform preferably consists of any existing engines and/or services plus a superposition of potential engines and potential services. When a suitable challenge (a "quantum challenge" seems a suitable name, for obvious reasons) is received by a platform, it preferably forces the platform to decide what states can actually be supported by the platform. The platform may make all actual states visible to the challenger. Alternatively, the challenged platform may just instantiate the challenged engine/service (if possible), and provide a conventional crypto-response back to the challenger to confirm provision of the engine/service associated with the label.

When a first platform wishes to verify that a target platform will provide a particular service, it sends a "quantum challenge" to the target platform. The TPM in the target platform inspects its (PCR) state measurements. If the desired state exists, either because the platform enters that state on receipt of the challenge or is already in that state, the TPM is preferably able to UNSEAL the appropriate label and preferably signs its response (including the label) using the platform identity

secret corresponding to that label. That is to say, it signs a nonce (received in the quantum challenge) plus the label corresponding to the desired service using the platform's identity secret (or the secret specifically allocated to that identity label). The first platform receives the response, checks the nonce and checks the label. The first platform then checks the signature, using the public key of a trusted Certification Authority, preferably the domain. This signature check may be done on-line, but is preferably done off-line using a public key loaded at manufacture or initialization. If all checks pass, the first platform believes that the target is running the particular service, and may decide to use or perform the service in cooperation with the target platform. Such a check may be all that is required if a verified platform identity and label provide sufficient trust. This may well be the case for a service that is internal to an organization and/or domain.

If payment but not authorization or identification is required to use or provide a service, and the requestor has money in an electronic form, the required amount of electronic money is preferably simply transferred between platforms at some point.

If payment is required to use or perform a service, the requestor is required to provide payment information. This preferably takes the form of a credit card number or credit card signature. A credit card number may belong to the platform or may belong to a user. If it belongs to the platform, the number is preferably stored in the platform, preferably in a secure manner, such as that provided by a TPCA TPM_SEAL command. If the credit card number belongs

to a user, it may be stored in the platform, preferably in a secure manner, and released after permission from the user. Alternatively, the user may provide the number to the platform. If a credit card signing secret belongs to the platform, the secret is stored in the platform, preferably in a secure manner such as that provided by a TPCA TPM_SEAL command, and used to sign a request. If a credit card secret belongs to the user, the secret may be stored in the platform and used to sign a request upon identification by the platform of the user, and/or of the user by the platform, and/or permission by the user. If a credit card secret belongs to a user, the secret may be stored in a token such as a smartcard, and used to sign a request upon identification by the platform of the user, and/or of the user by the platform. The protocol that must be used may be similar or the same as that used by a Point-Of-Sale terminal. The entity requesting payment then uses an existing credit agency, such as the VISA (Trade Mark) or Mastercard (Trade Mark) networks, to verify the payment authority.

If a user authentication is required, the same authentication is preferably used for all authorizations of that user. (Of course, multiple authentication/authorization values may be used, but this increases complexity.) Preferably this user authentication should also indicate the ability and desire to pay for the service. Note that, in this invention, the value that provides user authentication is not necessarily a true secret, merely a value that is guarded by the user and used in circumstances when the user wishes to give approval for some process, and is potentially visible to more entities than just the user and the verifier.

Preferably, therefore, this user authentication should have the properties of a credit card, whether a simple number (a value stored on a magnetic card, for example), or a signing secret (stored on a smartcard, for example).

If payment is not required but authorization and/or identification is required to use or perform a service, essentially the same process as for payment may be used. The difference being that the credit agency is used simply to verify the authorization/identification information, not to request payment. The credit agency might charge some money for providing this service. In effect, the credit agency is being paid for providing a PKI that would otherwise have to be provided by the domain. The protocol that must be used may be similar or the same as that used by a Point-Of-Sale terminal. Alternatively, a simpler protocol may be used. In the case of a card signing secret, the card might be requested to sign a nonce created by the domain and/or the credit agency, for example.

If payment is not required but authorization and/or identification is required to use or perform a service, a process similar to the same payment process may be used. The difference is that some identification and/or authorization server is used simply to verify the authorization/identification information, not to request payment.

When a platform is exhibiting the behaviour associated with a label, even just the root label, its software environment should preferably protect a user's data and applications from interference or prying by data and

applications belonging to other users. A person skilled in the art will be aware that compartments and virtual machines are well suited to this task. Information from one compartment or virtual machine is prevented from unauthorized interference or prying on other compartments or virtual machines.

When a platform is exhibiting the behaviour associated with a label, even just the root label, its software environment preferably includes a security service that cooperates with the mechanism providing the compartments and/or virtual machines. This security service preferably performs all the security services on behalf of the engines and services running in the compartments and/or virtual machines. The security service preferably verifies the labels of other platforms that are communicating with this platform. The security service may check that incoming messages have correct signatures that contain the correct label. The security service may cooperate with the platform's TPM to sign out-going messages while incorporating the correct label. The security service preferably manages the Diffie-Hellman protocol between platforms and sets up and takes down secure channels as required. The security service preferably manages all storage of secrets used to store data and/or applications belonging to another platform and/or user inbetween active sessions. The security service also preferably performs other security services, as described in a prior patent application "Performing a service on a computer" (referred to above), on behalf of the applications. The security service is preferably managed by an application on its host computer, but preferably accepts policies from remote platforms governing the data and applications and engine

and service for that remote platform on the local platform. The local management application preferably does not provide means to override those policies, but can refuse to execute processes on unacceptable terms.

One aspect of this invention is a modification that removes the differentiation between the client and the server, thus migrating the invention towards a peer-to-peer architecture. This aspect of the invention is that a client advertises its ability to participate in a particular service. The client preferably creates a signed certificate that states that the client is able to participate in a particular service. The certificate is preferably created by signing the label of the service using the identity of the client's TPM. The certificate preferably indicates that the client may be able to participate in the service, not that it wishes to, or is even able to, at the time that the certificate is posted. The client preferably posts that certificate to the advertising service. This process is analogous to the procedure followed by the server described above, where the server signs the label describing a service and posts the certificate to the advertising service, to indicate that the server might provide the service, not that the server will actually provide the service. The client preferably also posts to the advertising service an additional certificate, which is the public key of the client signed by a trusted domain, and is attestation by the domain that the client may be trusted to accurately state the services (labels) in which it may participate. Again, this mirrors the actions of the server described above.

Another aspect is a modification that provides a convenient revocation mechanism and permission mechanism. In the system described above, the server preferably posts to the advertising service the attestations by the domain that the server may be trusted to state the services (labels) in which they may participate. In this aspect of the invention, the domain preferably controls the advertising service and itself posts the attestation for both the client and the server. This permits the domain to dynamically control whether a particular computer is able to participate in a service. If the domain does not control the advertising service, the clients and servers preferably post attestations to the advertising service, but the domain preferably posts revocation certificates to the advertising service. Thus a computer, when visiting the advertising service, can discover whether a client and/or server might participate in a service, and whether the client and/or server is permitted to participate in a service. Both clients and servers preferably periodically visit the advertising service, and use the presence and/or absence of the attestation and revocation certificates to update their permissions to provide a participate in a given service.

All of the features described herein can be combined with any of the above aspects in any combination.

A specific embodiment of the present invention will now be described by way of example and with reference to the accompanying drawings, in which:

Figure 1 [gjp1] is a schematic diagram illustrating the information in a computing platform;

Figure 2 is a schematic diagram of the relationship between four computing platforms;

Figure 3 [gjp2]is a schematic illustration of a world wide web service page showing potentially available services;

Figure 4 is a flow chart illustrating the communication processes between first and second computing platforms;

Figure 5 is a flow chart illustrating the communication between third and fourth computing platforms;

Figure 6 is a schematic diagram of the architecture by which the second and fourth platforms provide a service/engine; and

Figure 7 is a flow chart illustrating the process by which a platform is maintained or upgraded.

Figure 1 illustrates the information that exists in a system consisting of platform-n 100, credential certificate 114, credential 116, credit card number 123 and credential 119, preferably either at manufacture or at initialization of the system. A TPM 101 in platform-n 100 contains a public key 102 of trusted domain-A. The TPM 101 also contains a statistically unique secret, which is a private (identity) key 103 of an asymmetric key pair, and used to prove the identity of the platform. A corresponding public (identity) key 112 is inside a credential (certificate) 114 that is signed 113 by trusted domain-A. The TPM 101 also contains labels 104,106 (but note that not all TPMs contain all labels). One label 104

indicates service-X. Another label 106 indicates engine-Y. The TPM 101 contains values 105 that correspond to measurements relevant to service-X. The TPM 101 contains values 107 that correspond to measurements relevant to engine-Y. Each label 104,106 of each engine/service is associated 108,109 with the set of values 105,107 that correspond to the measurements relevant to the engine/service. The platform-n 100 also contains programs 125. A set of programs 110 provides service-X. A set of programs 111 provides engine-Y. A set of values 105 associated with label 104 corresponds to the results of measurements that should be obtained if the platform 100 is executing the programs 110. The set of values 107 associated with label 106 corresponds to the results of measurements that should be obtained if the platform 100 is executing the programs 111. Each platform also contains all the programs 115 that are required to ensure that a platform is in the configuration that conforms to the meaning of a label. Label 104 is also inside credential 116 (in the form of an X.509 standard certificate) signed 118 by trusted domain-A and including a reference 117 to a description of service-X implied by label 104. Label 106 is also inside credential 119 (also an X.509 certificate) signed 121 by trusted domain-A and including a reference 120 to a description of engine-Y implied by label 106. A user 123 has either a credit card number 122 and/or a credit card signing-secret 124.

Figure 2 illustrates four platforms, platform-1 100, platform-2 200, platform-3 202, platform-4 203. Platform-1 100 contains TPM 101 and programs 125. Platform-2 200 contains a TPM 205 and programs 206. Platform-3 202 contains a TPM 207 and programs 208. Platform-4 203

contains a TPM 209 and programs 210. Each TPM 101,205,207,209 is capable of attesting that the associated platform 100,200,202,203 may be trusted to provide an engine and/or take part in a service. Also illustrated is an external credit verification service 204. A communication fabric 201(such as the Internet or a Local Area Network, for example) connects all four platforms and the credit verification service. The programs 125,206,208,210 in each platform 100,200,202,203 respectively provide at least a security service (e.g. 115 for platform-1 100 in Figure 1) plus applications to provide services (e.g. 110,111 for platform-1 100 also in Figure 1).

Figure 3 illustrates a Web service-page. Each platform (100, 200, 202 203) advertises (using a Web page) the engines and services that it may potentially provide. The information on this particular service page 301 advertises platforms that can provide service-X. One set 303 of information advertises the potential availability of service-X from platform-1 100. The information includes the certificate 309 of label-X 104 signed by platform-1, the certificate 114 of platform-1's public key signed by domain-A, and the IP address 302 of platform-1 100. Another set 306 of information advertises the potential availability of service-X from platform-2 200. The information includes the certificate 308 of label-X signed by platform-2, the certificate 304 of platform-2's public key signed by domain-A, and the IP address 305 of platform-2 200. Similar information 307 from other platforms may be available. The information includes the certificate 116 of label-X and its description, signed by domain-A.

Figure 4 is a flow chart that illustrates the process by which platform-1 100 uses service-X on platform-2 200 after consulting a web page 301. Column 401 illustrates events at platform-1 100. Column 402 illustrates events at platform-2 200. Column 403 illustrates events at the web server.

In step-1 404, platform-1 100 visits the Web service page 301. The service-page 301 is widely publicized and/or available at a well known address, for example.

In step-2 405, platform-1 100 verifies that the certificates 309,114,308,304,116 and certificates from other platforms are correctly signed.

In step-3 406, platform-1 100 discovers that platform-2 200 is able to provide service-X, since trusted domain-A has signed the public key of platform-2 in the certificate 304, trusted domain-A has signed the label X and description of service-X in certificate 116, and the private key of platform-2 has signed the label of service-X in certificate 308.

In step-4 407, platform-1 100 contacts platform-2 200 using IP address 305 and requests the provision of the service corresponding to the label X of service-X, using a challenge from platform-1 to platform-2.

In step-5 408, platform-2 200 attempts to start service-X using the programs 110. The attempt is successful, so the measurements taken by the TPM 205 in platform-2 200 now

indicate to the TPM 205 that it is permitted to use the label X 104 associated with service-X.

In step-6 409, platform-2 200 signs label X and the nonce 410 (from the incoming challenge) using its secret key, and sends the result 411 back to platform-1 100.

In step-7 412, platform-1 100 verifies the signature of platform-2 200 and hence verifies that platform-2 200 is now confirming the presence of service-X.

In step-8 415, platform-2 200 optionally performs a similar test on platform-1 100, to verify that platform-1 100 also contains service-X.

In step-9 416 , platform-1 100 and platform-2 200 cooperate to setup a secure channel, using the Diffie-Hellman protocol.

In step-10 417, platform-1 100 and platform-2 200 provide and consume service-X, by executing the service and passing data 418,419 signed by their respective private keys and including label X.

Figure 5 is a flow chart that illustrates the process by which platform-3 202 uses an engine-Y on platform-4 203. Column 501 illustrates events at a user of platform-3 202. Column 502 illustrates events at platform-3 202. Column 503 illustrates events at platform-4 203. Column 504 illustrates events at the web site. Column 505 illustrates events at the credit verification service 204. The protocol executes step 1 404 to step 9 416 of figure 4,

until step 10 417, because the use of engine-Y requires user authorization.

In step-10 507 of figure 5, platform 3 202 asks its user to permit the use of a credit card number associated with the user.

In step-11 508, the user grants permission by entering the credit card number into platform-3 202.

In step-12 509, platform-3 202 sends the credit card number to platform-4 203 over the secure channel set up in step 9 416 between platform-3 202 and platform-4 204.

In step-13 510, platform-4 203 contacts the credit agency 204 by a protocol outside the scope of this invention.

In step-14 511, the credit agency 204 confirms to platform-4 203 the validity of the credit card number. In this case, the credit card agency 204 charges platform-4 203 for this service, although it could just have easily have charged the owner of the credit card number.

Finally, in step-15 512, platform-3 202 and platform-4 203 provide and consume engine-Y, by instantiating the engine on platform-4 and executing data and applications provided by platform-3 on that engine. Data 513 passing back and forth is signed by their respective private keys and includes label Y.

Figure 6 illustrates the preferred architecture by which platform-2 provides service-X and platform-4 provides engine-Y. In this example, platform-2 200 and platform-4

203 execute service-X (provided by programs 1,2,3 110) in compartment-1 602 and execute engine-Y (provided by programs 1,6,8 111) in compartment-2 603. A security service 605 may also execute in a compartment 604. Other software processing 601, probably including kernel software, enforces and provides the compartment properties of the platforms 1 and 2. The security service 605 provides all security for service-X and engine-Y. The security service 605 intercepts messages 606,607 between the outside world and service-X. The security service 605 intercepts messages 606,608 between the outside world and engine-Y.

Figure 7 is a flow chart that illustrates the process by which a platform is maintained and/or upgraded. Column 701 illustrates events at domain-A. Column 702 illustrates events at the platforms in domain-A.

In step-1 703, domain-A signs new values that are to be associated with existing label-n, signs a new set of {label-p and its associated values} and signs new programs.

In step-2 704, domain-A sends those new data to all of the platforms in its domain.

In step-3 705, each platform's TPM 101, 205, 207, 209 verifies the signatures on the incoming data using the public key of domain-A that was stored in each TPM. In step-4 706, the programs are stored in the platform, the values associated with label-n in the TPM 101, 205, 207, 209 are replaced by the new values, and the new set of

{label-p and its associated values} is installed in the TPM 101, 205, 207, 209.

The previous description assumes that a single domain both controlled the platforms and provided/managed the services. This does not have to be the case. A viable alternative is for one domain to control the platforms and a different domain to define/provide the services. Then one important change is that a platform, when using a label, should sign and provide a statement of the interpretation of the label by the platform. This is necessary in order that a second platform can reliably deduce the implications at a first platform of a particular label at that first platform. Platform -1 could provide this statement, for example, by modifying the credential 308 (currently a label signed by platform-1) so that the credential 308 includes the precise meaning of the label as implemented by platform-1. This might be done, for example, by modifying the credential 308 to include a reference to another credential 116, which is a signed description of the meaning of a label.

If different domains provide the platforms and the services, a service provided by the platform domain may be used to install and update the foreign services in the platforms.

If different domains provide the platforms and the services, an engine service provided by the platform domain may be used to provide services in the platforms on behalf of the service domain.

The system and protocol described herein provides significant advantages because the security requirements are considerably reduced and the total number of secrets in a system is reduced, thus giving advantageous simplification of the security system. Also the security infrastructure is kept out of the applications and engines to improve their efficiency.

The computing platforms described herein have been explained with particular reference to the TCPA specification. However, that specification is just an example of a trusted component of a computing platform and other types of trusted component are equally applicable to the invention described herein.

Claims

1. A computer system comprising at least one platform containing a trusted entity and at least one label, the trusted entity being operable such that use of the or each label by the trusted entity is dependent on the presence or potential presence of a predetermined software state in the or each platform.

2. A computer system as claimed in claim 1 in which the at least one label is adapted to indicate or advertise the presence or potential presence of the predetermined software state in the or each platform.

3. A computer system as claimed in claim 1, in which the predetermined software state includes a particular configuration of computing resources and/or software described directly or indirectly by the or each label.

4. A computer system as claimed in claim 3, in which the or each label describes a service which can potentially be offered by the at least one platform.

5. A computer system as claimed in claim 1, in which the labels in at least two platforms are the same where the labels describe essentially the same configuration of computing resources and/or software.

6. A computer system as claimed in claim 5, in which the labels in the two platforms are essentially the same where the labels describe a particular configuration of

computing resources and/or software related to the same distributed computing engine or distributed service.

7. A computer system as claimed in claim 1 in which the or each label is widely published and one form of published label is signed using a secret known to the platform.

8. A computer system as claimed in claim 1, in which the or each label is widely published and one form of published label includes descriptive information and is signed by a trusted entity.

9. A computer system as claimed in claim 1, in which the or each label is widely published and one form of published label includes descriptive information about the configuration of computing resources and/or software associated with the label and is signed by a trusted entity.

10. A computer system as claimed in claim 1, in which the or each label is widely published and one form of published label includes an offer to provide a configuration of computing resources and/or software associated with the label.

11. A computer system as claimed in claim 10, in which the or each label is signed using a secret known to the platform.

12. A computer system as claimed in claim 1, in which the reception by the platform of a cryptographic challenge incorporating one of said at least one labels from a

second platform causes the platform to determine whether the computing resources and/or software associated with said label can be provided by the platform.

13. A computer system as claimed in claim 1, in which proof of possession of a label by a platform is sufficient for another entity to cooperate with that platform for the purposes of using and/or providing the computing resources and/or software described by that label.

14. A computer system as claimed in claim 3, operable such that the right to use the computing resources and/or software described by the label depends on provision of one or more of:

proof of possession of a platform secret,
proof of possession of a user secret,
presentation of a non-secret authorisation value associated with a user whose use is known to be indicative of a request from the user,
presentation of a non-secret authorisation value associated with a user whose use is known to be indicative of agreement by the user to tender payment.

15. A computer system as claimed in claim 1, in which at least one platform contains trustworthy integrated mandatory enforcement controls and security capabilities that transparently provide security and privacy to applications that are at least substantially ignorant of security and privacy, and requires permission from at least one other platform to permit the flow of information to the resources allocated to said other platform from the resources allocated to the first-mentioned platform.

16. A computer system comprising at least one platform containing a trusted entity and at least one label, the trusted entity being operable such that use of the or each label by the trusted entity is dependent on the presence or potential presence of a predetermined software state in the or each platform, wherein the at least one label is operable to indicate or advertise the presence or potential presence of the predetermined software state in the or each platform, and wherein the or each label is widely published and describes a service or resource which can potentially be offered by the at least one platform.

17. A computer system comprising at least one platform containing a trusted entity and at least one label, wherein the label describes a predetermined software state in the or each platform and wherein the trusted entity is operable to use the label if the predetermined software state is described by the label is present or potentially present in the or each platform.

18. A computer system as claimed in claim 17, in which the trusted entity will sign the at least one label with a secret known to the platform only if the predetermined software state is present or potentially present in the at least one platform.

19. A computer system as claimed in claim 17, in which the at least one label publicly discloses the predetermined software state in order to indicate the availability of a service or the resource on the or each platform.

20. A computer system comprising at least one platform containing a trusted entity and at least one application, wherein the platform is operable to perform security functions for the computer system.

21. A computer system as claimed in claim 20, in which the platform performs substantially all security functions and the applications perform substantially no security functions.

22. A computer system as claimed in claim 20, in which the platform is operable to apply mandatory security controls on communications from the computer system.

23. A computer system as claimed in claim 20, in which updates of security functions are broadcast across the system to the at least one platform.

24. A method for a computer system to signal the potential availability of a computing resource or service comprises providing a platform containing a trusted entity with at least one label, wherein the label is used by the platform only when a predetermined software state is present or potentially present in the platform.

25. A method as claimed in claim 24, wherein the label describes the computing resource or service, which is defined by the predetermined software state.

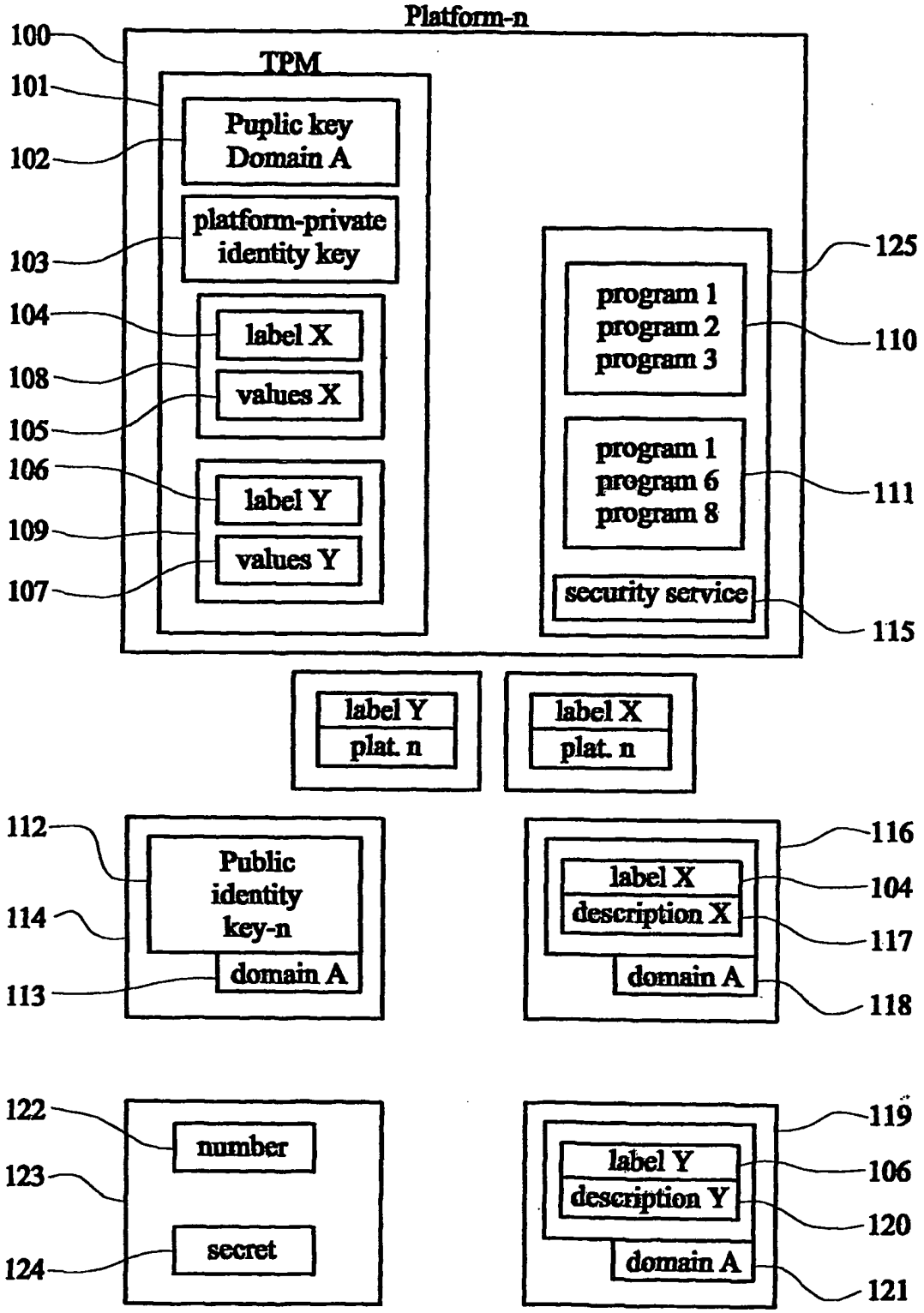

Platform-n
100
TPM
101
Puplic key Domain A
102
platform-private identity key
103
104
label X
108
values X
105
106
label Y
109
values Y
107
125
program 1
program 2
program 3
110
program 1
program 6
program 8
111
security service
115
label Y
plat. n
label X
plat. n
112
Public identity key-n
114
domain A
113
116
label X
104
description X
117
domain A
118
122
number
123
secret
124
119
label Y
106
description Y
120
domain A
121

FIG. 1

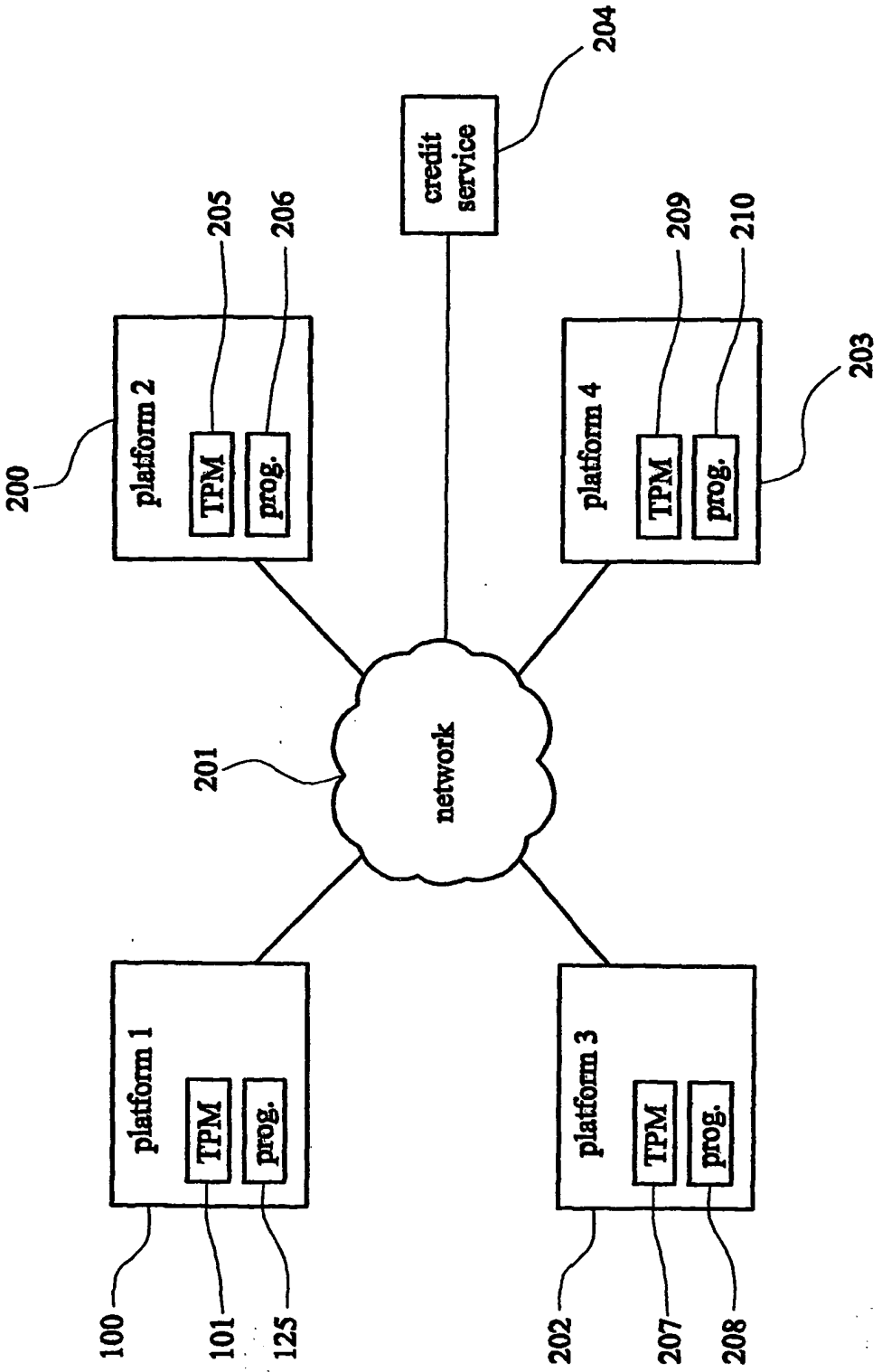

credit service
204
205
206
209
210
203
200
platform 2
TPM
prog.
platform 4
TPM
prog.
network
201
platform 1
TPM
prog.
platform 3
TPM
prog.
100
101
125
202
207
208

FIG. 2

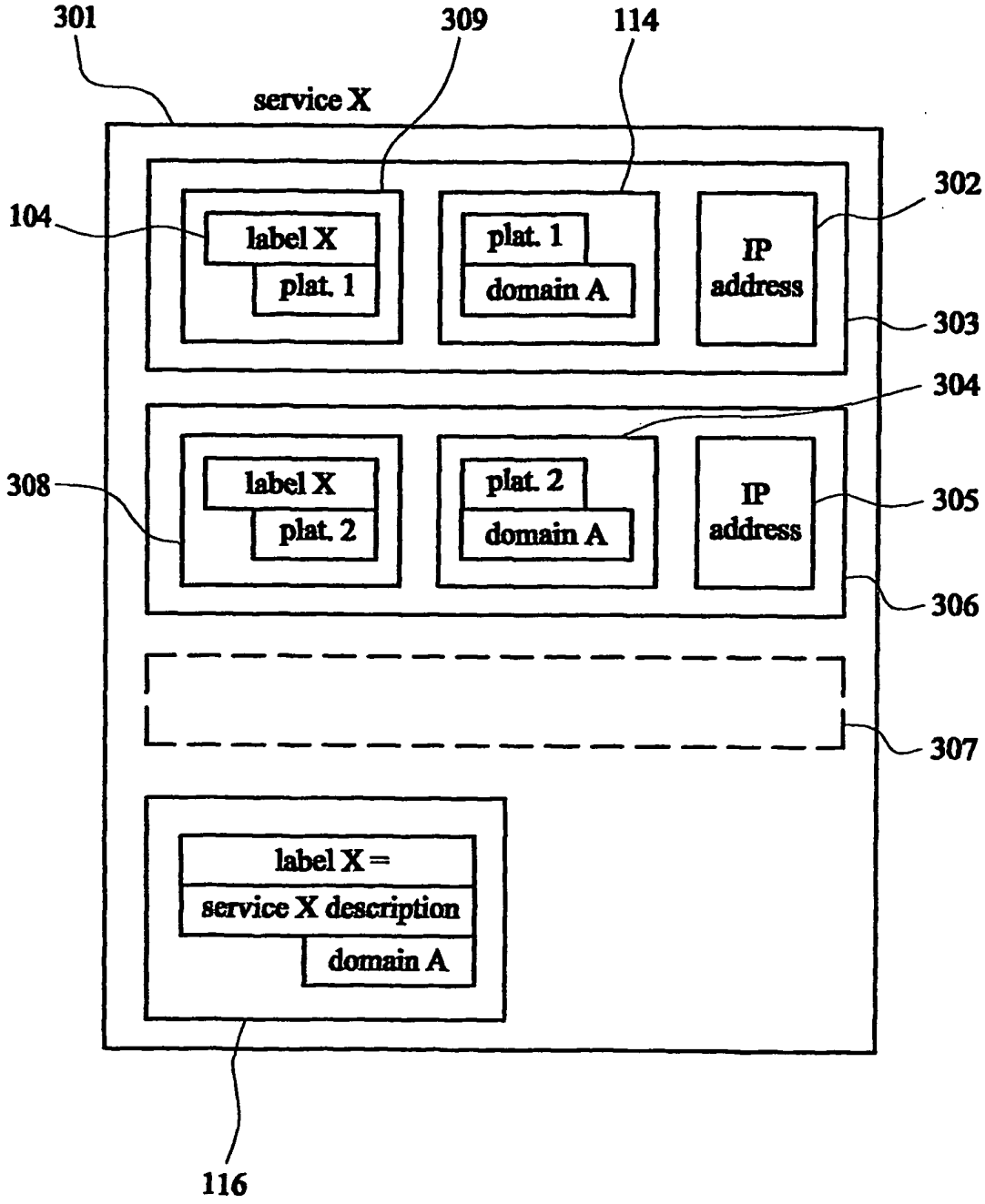

301
309
114
service X
104
label X
plat. 1
plat. 1
domain A
IP address
302
303
304
308
label X
plat. 2
plat. 2
domain A
IP address
305
306
307
label X =
service X description
domain A
116

FIG. 3

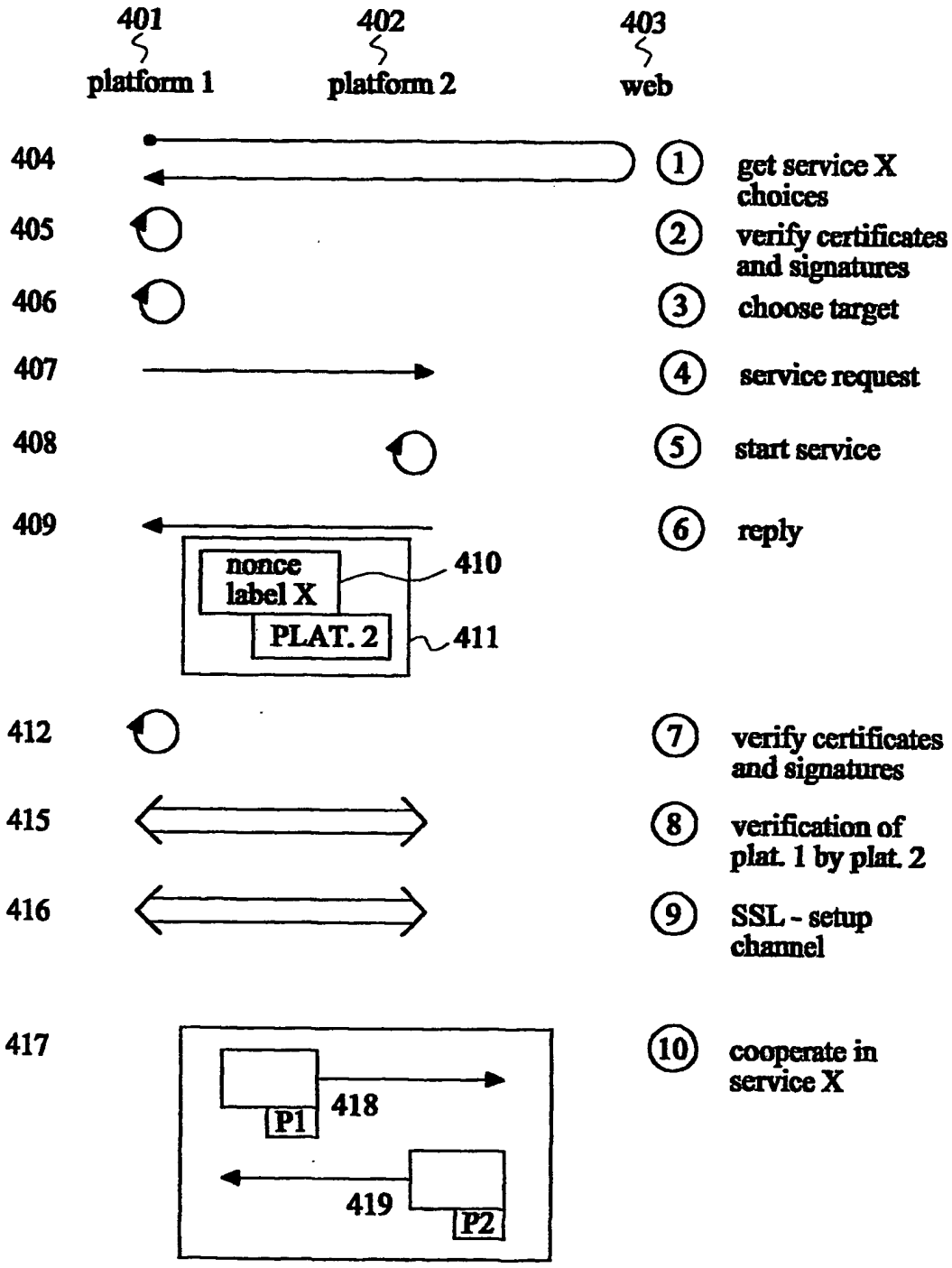

401
platform 1
402
platform 2
403
web
404
1
get service X choices
405
2
verify certificates and signatures
406
3
choose target
407
4
service request
408
5
start service
409
6
reply
nonce
label X
410
PLAT. 2
411
412
7
verify certificates and signatures
415
8
verification of plat. 1 by plat. 2
416
9
SSL - setup channel
417
10
cooperate in service X
P1
418
419
P2

FIG. 4

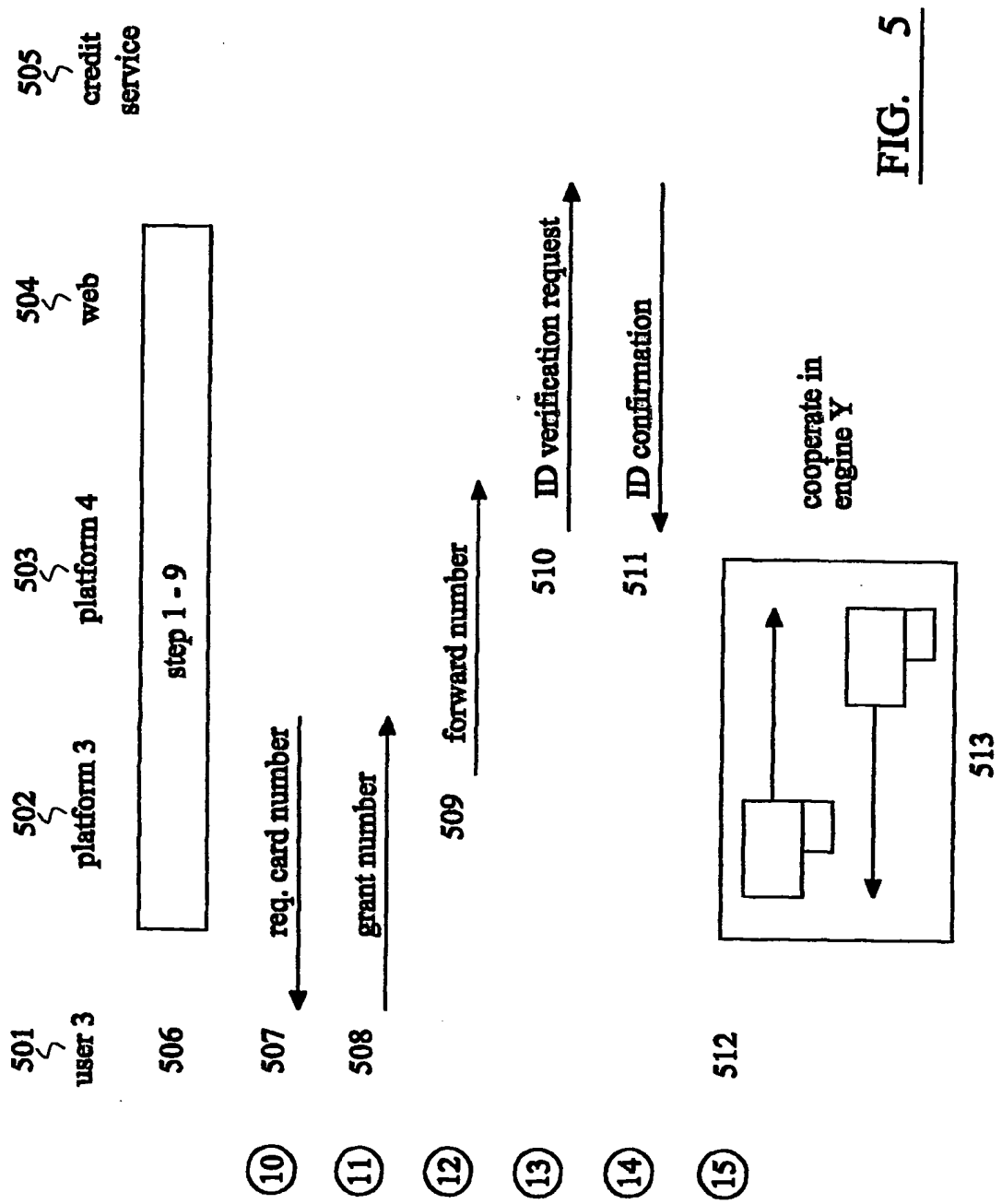

501
user 3
502
platform 3
503
platform 4
504
web
505
credit service
506
step 1 - 9
507
req. card number
508
grant number
509
forward number
510
ID verification request
511
ID confirmation
512
cooperate in engine Y
513
10
11
12
13
14
15

FIG. 5

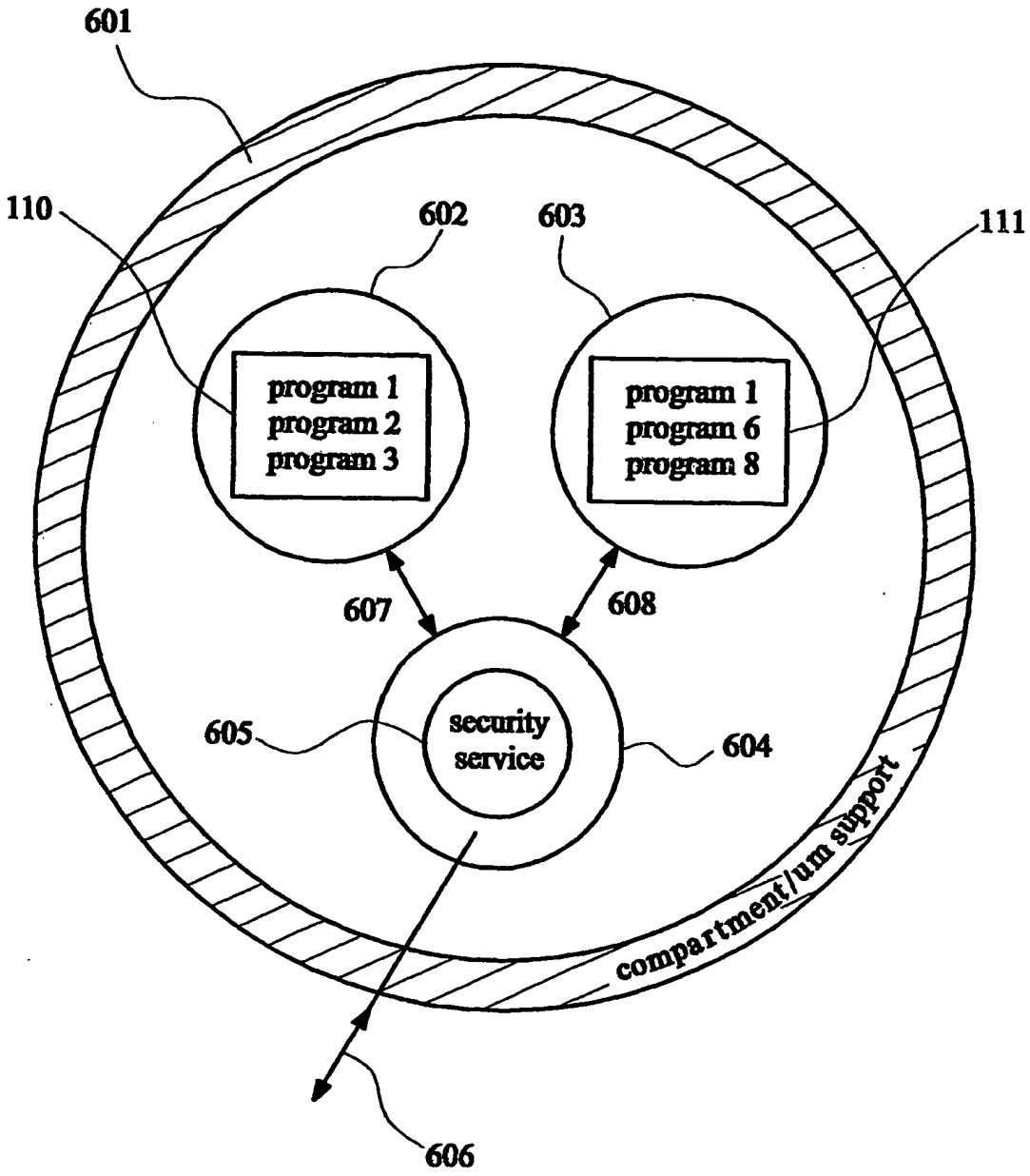

601
110
602
603
111
program 1
program 2
program 3
program 1
program 6
program 8
607
608
security service
605
604
compartment/um support
606

FIG. 6

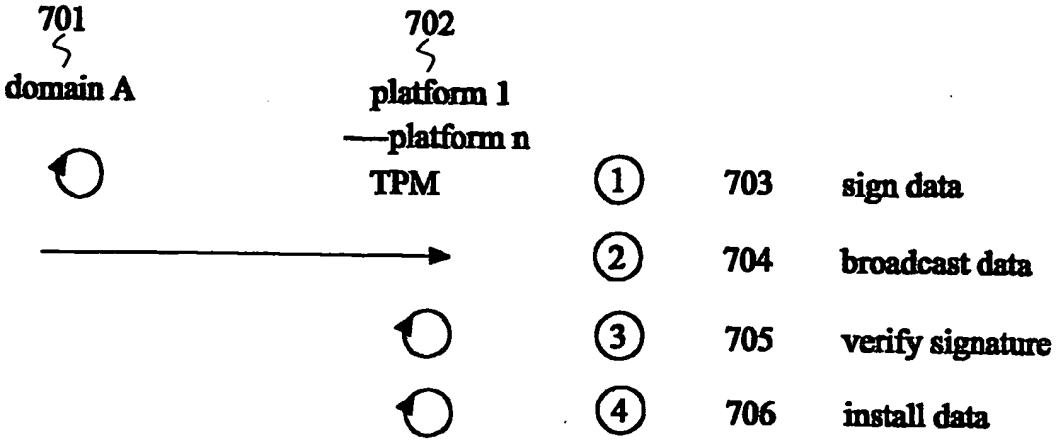

701
domain A
702
platform 1
—platform n
TPM
1
703
sign data
2
704
broadcast data
3
705
verify signature
4
706
install data

FIG. 7